60. ročník Matematické olympiády na středních školách

Kategorie P

In: Zdeněk Dvořák (editor); Zbyněk Falt (editor); Karel Horák (editor); Peter Novotný (editor); Martin Panák (editor); Jaromír Šimša (editor); Jaroslav Švrček (editor); Pavel Töpfer (editor): 60. ročník Matematické olympiády na středních školách. Zpráva o řešení úloh ze soutěže konané ve školním roce 2010/2011. 52. Mezinárodní matematická olympiáda. 5. Středoevropská matematická olympiáda. 23. Mezinárodní olympiáda v informatice. (Czech). Olomouc: Univerzita Palackého v Olomouci, 2015. pp. 100–124.

Persistent URL: http://dml.cz/dmlcz/405215

Kategorie P

Texty úloh

P – I – 1

Indiana a poklad

Po dlouhé a nebezpečné cestě plné dobrodružství se konečně před Indianou rozprostřel pohled na starodávné pohřebiště aztéckých králů. Byl to hrob vedle hrobu podél dlouhé cesty, některé se lišily výškou, ale jinak byly stejné. Na prvním kameni byl následující nápis:

Jsi-li moudrý, pochopíš,
kde můj poklad našel skrýš.
Se zlou však se potáže,
kdo špatný hrob ukáže.

Výklad těchto veršů je prostý: pokud otevřete hrob s pokladem, tak tento poklad získáte, v opačném případě nezískáte nic a nejspíš vás něco rozmáčkne. Verše naštěstí pokračovaly:

Kdo volí z prostředních, neprohloupí,
pokud si nakonec největší koupí.

No a teď už je jistě jasné i vám, kde se poklad nachází. Stačí jen najít největší z prostředních výšek hrobů a poklad bude objeven.

Soutěžní úloha. Je zadána posloupnost N kladných celých čísel v_1, $v_2, \ldots, v_N$ a celé liché číslo K. Vaším úkolem je najít maximum ze všech mediánů souvislých podposloupností délky K. Medián získáte pro danou podposloupnost tak, že všechna její čísla seřadíte podle velikosti a zvolíte prostřední prvek (tzn. $\frac{1}{2}(K+1)$-tý prvek). Například pro posloupnost výšek hrobů 4, 5, 1, 3, 2, 4 a $K = 3$ tak získáte posloupnost mediánů 4, 3, 2, 3, a hledaným výsledkem je tedy číslo 4.

Formát vstupu: Program načte vstupní data ze standardního vstupu. První řádek vstupu obsahuje dvě celá čísla N a K, počet hrobů a délku uvažovaných podposloupností, $1 \leqq N \leqq 1\,000\,000$ a $1 \leqq K \leqq 10\,000$.

Každý z následujících N řádků obsahuje výšku jednoho z hrobů (v pořadí $v_1, v_2, \ldots, v_N$). Výšky jsou přirozená čísla menší než 1 000 000 000. Můžete předpokládat, že pro 40 % vstupů platí $K < 100$.

Formát výstupu: Na standardní výstup vypište jedno číslo, které je rovno maximu ze všech mediánů souvislých úseků délky K v posloupnosti čísel zadaných na vstupu.

Příklad:

Vstup:	*Výstup:*
6 3 4 5 1 3 2 4	4

P – I – 2

Poklad podruhé

Poté co Indiana našel největší z prostředních hrobů, zjistil, že nápis nebyl úplně přesný. Místo pokladu však byl pod kamenem jen vchod do další chodby. Ta byla vydlážděna čtvercovými dlaždicemi a hned na první dlaždici byl vyrytý tento nápis:

Šlápneš-li na každou z nás právě jednou,
Tvé ruce nad hlavu poklady zvednou.
Tou hlavou však zaplatit musíš,
když sestru s lebkou poškádlit zkusíš.

Chodba měla na šířku přesně 3 dlaždice a dlouhá byla, kam až oko dohlédlo. Na některých z dlaždic byly namalované lebky, na jiných dlaždicích pak ležely skutečné lebky (pravděpodobně předchozích archeologů).

Indiana velmi rychle nápis pochopil — musí na každou dlaždici (kromě těch zakázaných) šlápnout právě jednou, jen tak vede cesta k pokladu. V tu chvíli ale začal litovat toho, že má tak velké nohy. Ať se snažil, jak chtěl, nikdy se mu nepodařilo stoupnout jenom na jednu dlaždici, ale vždycky stoupnul na dvě sousední. To úkol samozřejmě zkomplikovalo.

Soutěžní úloha. Na vstupu je dána délka chodby N, tzn. chodbu tvoří $3 \times N$ dlaždic. Z těchto dlaždic je K zakázaných, na které Indiana nesmí

vstoupit. Vaším úkolem je určit, kolika způsoby lze tuto chodbu pokrýt stopami velikosti 1×2 dlaždice tak, aby každá nezakázaná dlaždice byla pokryta právě jednou stopou a žádná zakázaná dlaždice nebyla pokryta. Protože výsledné číslo může být velmi velké, vypište zbytek po dělení tohoto čísla zadaným číslem L.

Formát vstupu: Program načte vstupní data ze standardního vstupu. Na prvním řádku jsou zadána přirozená čísla N, K a L, $1 \leqq N \leqq$ $\leqq 10\,000\,000$, $1 \leqq K \leqq \min(3N - 1, 1\,000\,000)$ a $2 \leqq L \leqq 1\,000\,000$. Následuje K řádků, přičemž na každém z nich je dvojice čísel X_i a Y_i, $1 \leqq X_i \leqq 3$ a $1 \leqq Y_i \leqq N$, které udávají souřadnice i-té zakázané dlaždice. Vždy bude platit, že $Y_1 \leqq Y_2 \leqq \ldots \leqq Y_K$.

Můžete předpokládat, že 20 % vstupů bude splňovat $1 \leqq N \leqq 40$.

Formát výstupu: Na standardní výstup vypište jediný řádek, který obsahuje počet možností, kolika způsoby lze chodbu pokrýt. Toto číslo je uvedeno modulo L. Všimněte si, že pro některé vstupy nemusí existovat žádné řešení — v takovém případě vypište nulu.

Příklad:

Vstup:	*Výstup:*
`5 3 13`	`3`
`3 1`	
`1 2`	
`2 4`	

P – I – 3

Řeka

Za devatero horami se nachází rozsáhlý prales, kterým protéká dlouhá řeka. I přes svou obrovskou délku nemá řeka žádné přítoky a ani se nikde nerozvětvuje.

V pralese roste mnoho vzácných druhů stromů, a proto není divu, že u řeky leží dřevorubecké tábory. Vždy, když dřevorubci pokácí dostatečné množství stromů, sestaví z nich vor a pošlou ho dolů po řece.

Kromě dřevorubeckých táborů se u řeky také nacházejí pily. Zaměstnanci každé pily sledují řeku, a když k nim dorazí vor, odchytí ho a všechno dřevo spotřebují. Občas je pila zavřená kvůli údržbě, v té době ignoruje vory a nechává je plout dále po řece.

Zaměstnancům pil vadí, že nevědí dopředu, kdy mají očekávat dodávku dřeva a zbytečně tráví čas pozorováním řeky. Pomozte jim a na-

pište program, který jim bude posílat upozornění na blížící se zásilku dřeva.

Soutěžní úloha. Řeka má délku N kilometrů a pozice bodů na ní budeme označovat vzdáleností od pramene. V každém z bodů $1, \ldots, N$ se nachází buď dřevorubecký tábor, nebo pila. Umístění pil na řece je dáno na začátku výpočtu. Víte, že pil je poměrně mnoho vzhledem k délce řeky; můžete předpokládat, že počet pil P je řádově stejný jako délka řeky.

Váš program bude dostávat události následujícího typu: „pila v bodě b zahajuje údržbu“, „pila v bodě b je opět v provozu“ a „z tábora v bodě t byl odeslán vor“. Pro každý odeslaný vor váš program odpoví, ve které pile bude zpracován. Předpokládejte, že řeka teče natolik rychle, aby mezi odesláním voru a jeho zachycením nebyla v žádné pile zahájena ani ukončena údržba.

Formát vstupu: Program načte vstupní data ze standardního vstupu. První řádek obsahuje přirozená čísla N, P a M, udávající délku řeky, počet pil na řece a počet událostí ($1 \leqq N \leqq 100\,000$, $1 \leqq P \leqq N$ a $1 \leqq \leqq M \leqq 1\,000\,000$). Na následujících P řádcích jsou popsány polohy pil: na i-tém z těchto řádků se nachází číslo n_i ($1 \leqq n_i \leqq N$), udávající, že i-tá pila se nachází ve vzdálenosti n_i od pramene řeky. Můžete předpokládat, že $1 \leqq n_1 \leqq \ldots \leqq n_P \leqq N$.

Dále následuje M řádků popisujících události v chronologickém pořadí. Na každém z těchto následujících řádků se nachází popis jedné události:

- ▷ písmeno `U` následované mezerou a přirozeným číslem d ($1 \leqq d \leqq \leqq N$) — pila ve vzdálenosti d od pramene zahajuje údržbu. Můžete
- ▷ předpokládat, že číslo d je jedno z čísel $n_1, \ldots, n_P$.
  písmeno `K` následované mezerou a přirozeným číslem d ($1 \leqq d \leqq \leqq N$) — pila ve vzdálenosti d od pramene končí údržbu. Můžete
- ▷ předpokládat, že číslo d je jedno z čísel $n_1, \ldots, n_P$.
  písmeno `V` následované mezerou a přirozeným číslem t ($1 \leqq t \leqq \leqq N$) — z tábora ve vzdálenosti t od pramene řeky je odeslán vor. Můžete předpokládat, že t je odlišné od všech čísel $n_1, \ldots, n_P$.

Formát výstupu: Program vypíše na standardní výstup tolik řádků, kolik vorů bylo celkem odesláno. Každý řádek bude obsahovat jedno celé číslo, které udává vzdálenost pily, která zpracuje vor, od pramene řeky. Pokud vor nebude zpracován žádnou pilou na řece, vypište 0.

P – I – 4

Grafový počítač

V tomto ročníku olympiády budeme pracovat se speciálním grafovým počítačem. V následujícím studijním textu je popsáno, jak takový počítač funguje a jak se programuje.

Běžné počítače počítají s čísly. Železniční inženýři v Tazmánii si jednoho dne všimli, že většina problémů, které potřebují řešit, se týká grafů. Proto během jedné polední přestávky vynalezli grafovou jednotku, která umí provádět všechny běžné operace s grafy, a to dokonce v konstantním čase. Sice zatím nevymysleli, jak ji sestrojit, ale i tak si můžeme v tomto ročníku olympiády vyzkoušet, jak se na takovém *grafovém počítači* programuje.

Příklad:

Vstup:	*Výstup:*
`6 3 9`	`2`
`2`	`4`
`4`	`6`
`6`	`4`
`V 1`	`2`
`V 3`	`0`
`V 5`	
`U 2`	
`V 1`	
`K 2`	
`V 1`	
`U 6`	
`V 5`	

Nejdříve definujme, s čím grafový počítač pracuje.

Graf si můžeme představovat třeba jako body v rovině (těm budeme říkat *vrcholy* grafu), jejichž některé dvojice jsou spojeny hranou. Může to tedy třeba být mapa železniční sítě: vrcholy jsou zastávky, dvě zastávky jsou spojeny hranou, pokud mezi nimi vede přímá trať. Pokud se hrany kříží, předpokládáme, že se jedná o mimoúrovňová křížení.

Řečeno formálně, graf je dvojice (V, E) taková, že V je libovolná konečná množina (jejím prvkům se říká vrcholy) a E je množina neuspořádaných dvojic prvků z V (tedy hran).

Upřesněme ještě, že mezi dvěma různými vrcholy může vést maximálně jedna hrana a že nejsou povoleny hrany, jejichž oběma konci je tentýž vrchol.

Dále ke grafu můžeme přidat *ohodnocení*. Vrcholům a hranám může být přiřazeno nezáporné celé číslo. V případě hran může znamenat například délku kolejí, v případě vrcholů může popisovat mýtné, které se platí za průjezd. Někdy jím také můžeme značit různé vlastnosti: například u našeho železničního příkladu může být vrchol odpovídající stanici ohodnocen jedničkou, zatímco vrcholy v zastávkách dvojkou.

Reprezentace grafu

Grafový počítač ukládá grafy tak, že vrcholy jsou určeny přirozenými čísly od 1 do počtu vrcholů. Těmto číslům budeme říkat *identifikátory* (zkráceně *id*) vrcholů.

Hrany budeme vždy identifikovat pomocí čísel vrcholů, které hrana spojuje.

Každý vrchol a každá hrana mají své ohodnocení. To má buď hodnotu nezáporného celého čísla nebo speciální hodnotu `undef` (tzn. nedefinováno). Aby se to nepletlo, budeme číslům na hranách říkat *váhy hran,* zatímco těm ve vrcholech *značky vrcholů.*

K programování grafového počítače použijeme běžný programovací jazyk, například Pascal nebo C, který rozšíříme o několik datových typů a funkcí. Zde je budeme ukazovat v syntaxi Pascalu, v C budou obdobné.

Datové typy

- Typ `Graph` — do tohoto typu se dá uložit jeden (celý, libovolně velký) graf. Mezi proměnnými a hodnotami tohoto typu funguje obvyklé přiřazování a porovnávání na rovnost.
- Typ `Value` popisuje ohodnocení vrcholu nebo hrany. Lze do něj ukládat nezáporná celá čísla a konstantu `undef`. Hodnoty tohoto typu různé od `undef` jsou kompatibilní s pascalským typem `Integer`, v případě jazyka C s typem `int`.

Operace se strukturou grafu

- Konstanta `EmptyG`. V této konstantě je uložen prázdný graf. To je takový, který nemá žádné vrcholy (tedy ani hrany).
- Funkce `AddV(G,z)` přidá do grafu G nový vrchol ohodnocený značkou z. Do přidaného vrcholu zatím nevedou žádné hrany. Nový vrchol bude zařazen jako poslední, tedy jeho *id* bude nejvyšší. Funkce vrací toto *id*.
- Procedura `DelV(G,id)` smaže vrchol s daným *id*. Zbývající vrcholy „srazí doleva“, aby nevznikla díra (tedy z $id+1$ se stane id, z $id+2$ se stane $id+1$ atd.). Zároveň odstraní všechny hrany, které končily ve smazaném vrcholu.
- Procedura `AddE(G,x,y,w)` vytvoří hranu mezi vrcholy s *id* x a y, ohodnocenou vahou w. Hrana nesmí před voláním této procedury existovat.
- Procedura `DelE(G,x,y)` odstraní hranu mezi vrcholy x a y (nesmí být volána, pokud hrana neexistuje).
- Funkce `TestE(G,x,y)` zjistí, jestli mezi vrcholy x a y vede hrana.

Manipulace s ohodnocením

- ▷ Funkce `GetV(G,id)` vrací značku zadaného vrcholu.
- ▷ Procedura `SetV(G,id,z)` nastaví značku zadaného vrcholu.
- ▷ Procedura `SetAllV(G,z)` ji nastaví všem vrcholům grafu.
- ▷ Procedura `ReplaceV(G,zold,znew)` všem vrcholům, které měly značku *zold*, ji změní na *znew*.
- ▷ Obdobně fungují `GetE(G,x,y)`, `SetE(G,x,y,w)`, `SetAllE(G,w)` a `ReplaceE(G,wold,wnew)`. Pracují s vahami hran místo značek vrcholů. Pro identifikaci hrany se používají *id* vrcholů x a y, mezi kterými vede. Procedura `SetE` hranu založí, pokud ještě neexistuje.

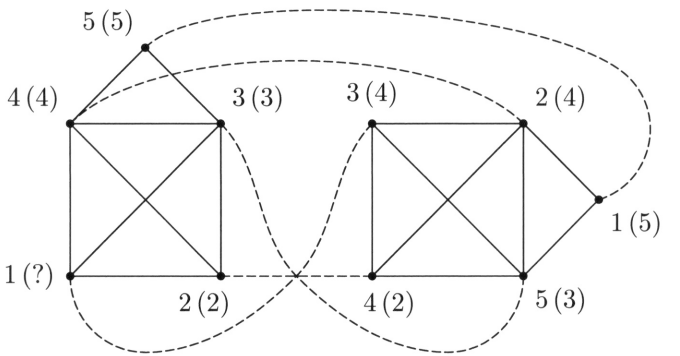
Statistické funkce

- ▷ Funkce `CountV(G)` odpoví, kolik vrcholů se nachází v grafu.
- ▷ Funkce `SumV(G)` vrací součet značek všech vrcholů, přičemž `undef` se počítá jako 0. Pokud graf nemá žádné vrcholy, vrací 0.
- ▷ Funkce `CountE(G)` a `SumE(G)` fungují obdobně pro hrany a jejich váhy.

Globální operace

- ▷ Funkce `Iso(G,H,veq,eeq)` zjistí, jestli jsou grafy G a H *isomorfní*. Isomorfismem myslíme, že lze jednomu z grafů přečíslovat vrcholy tak, aby se shodoval s druhým grafem. Dva grafy jsou shodné, pokud mají stejné množiny vrcholů i hran; navíc se jim musí shodovat značky vrcholů a váhy hran podle toho, jak určují parametry *veq* (pro vrcholy) a *eeq* (pro hrany). Tyto parametry mohou nabývat následujících hodnot:
  - ∗ `any` — libovolné dva vrcholy/hrany se rovnají (na ohodnocení se nehledí).
  - ∗ `value` — odpovídající si vrcholy/hrany musejí mít stejné ohodnocení. Hodnotu `undef` ale považujeme za „žolíka“, který se rovná libovolné hodnotě.
  - ∗ `value_strict` — vrcholy/hrany musejí mít stejné ohodnocení, `undef` se rovná jen `undef`u.
  - ∗ `value_defined` — vrcholy/hrany musejí mít stejné ohodnocení, ale `undef` se nerovná ničemu, ani `undef`u.
  - ∗ `id` — vrcholy musejí mít stejná *id* (toto lze aplikovat jen na vrcholy, neboť hrany nemají *id*). Jinými slovy, zakazujeme přečíslovávat vrcholy, ale na jejich ohodnocení nehledíme.
  - ∗ `none` — žádné dva vrcholy/hrany nejsou identické. Ač to vypadá neužitečně, tuto možnost použijeme v dalších funkcích.

Jak isomorfismus funguje, je vidět na následujícím obrázku. Čísla před závorkami jsou *id* vrcholů, v závorkách jejich značky (otazník značí `undef`). Všechny hrany mají váhu `undef`. Grafy jsou isomorfní (čárkované čáry ukazují, který vrchol odpovídá kterému), pokud *veq* nastavíme na `value` nebo `any` a *eeq* na `any`, `value` nebo `value_strict`. V ostatních případech isomorfní nejsou.

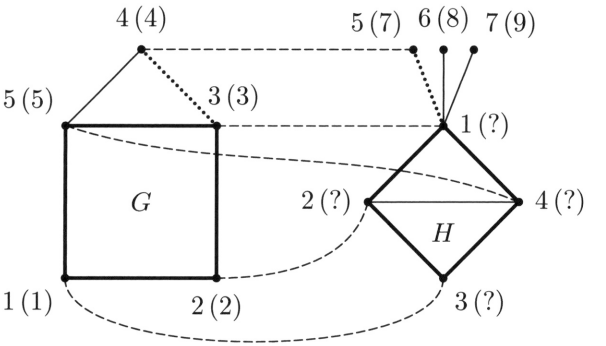


▷ Funkce `Find(G,H,veq,eeq)` najde podgraf grafu G (tedy takový graf, který lze získat z G odstraněním některých vrcholů a hran) isomorfní s grafem H. Výsledkem funkce bude tento podgraf, přičemž vrcholy budou očíslované podle grafu H a ohodnocení vrcholů a hran bude pocházet z grafu G. Pokud hledaný podgraf neexistuje, funkce vrátí `EmptyG`. Parametry *veq* a *eeq* určují stejně jako u funkce `Iso`, jak se chová isomorfismus.

Pokud existuje více isomorfních podgrafů, funkce `Find` nalezne nejlehčí z nich (takový, který má nejmenší součet vah hran, jak by ho spočítala funkce `SumE`). Pokud i tak existuje více řešení, `Find` vrátí libovolné z nich.

▷ Funkce `Common(G,H,veq,eeq)` najde největší společný podgraf grafů G a H. Přesněji, najde graf, který je isomorfní (podle *veq* a *eeq*) s některým podgrafem G i některým podgrafem H. Ze všech možných řešení si navíc vybere takové, které má největší možný počet vrcholů, a z takových pak to s největším počtem hran. Pokud i těch je více, vybere si libovolně.

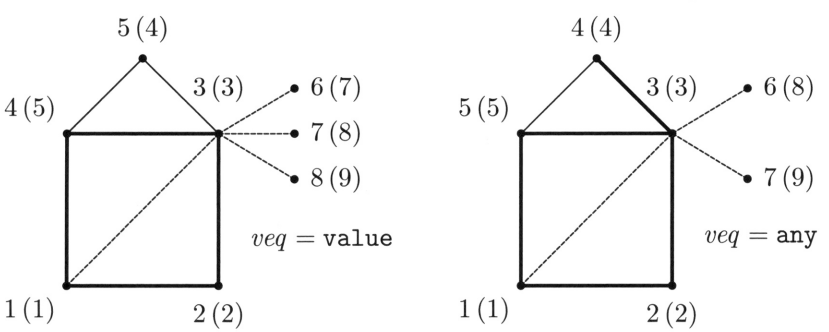
Výsledný graf bude mít *id* vrcholů ve stejném pořadí, jako je měl odpovídající podgraf v G (jen „sražená k sobě“). Ohodnocení vrcholů a hran bude také zděděno z grafu G.



Dvojice grafů na předchozím obrázku má největší společný podgraf při *veq* = `value` obtažený tučně. Čárkovaně je naznačeno jedno z možných přiřazení vrcholů. Při *veq* = `any` přibude do společné části ještě vrchol 5 a tečkovaná hrana $\{3,4\}$ nalevo, která může odpovídat kterékoli z hran $\{1,5\}$, $\{1,6\}$ a $\{1,7\}$ napravo.

▷ Funkce `Join(G,H,veq,eeq)` sloučí grafy G a H. Můžete si to představit tak, že je „slepí za jejich největší společný podgraf." Udělá to tak, že nejprve nalezne největší společný podgraf (tak jako ve funkci `Common`), pak k němu doplní zbývající vrcholy grafu G a nakonec vrcholy grafu H (*id* vrcholů výsledného grafu tedy budou v tomto pořadí). Hrany, váhy a značky přitom zdědí z obou grafů, přičemž pokud se nějaký vrchol nebo hrana vyskytují v obou grafech, řídí se ohodnocením z grafu G.

Join grafů z předchozího obrázku vypadá následovně:



Tučně je vyznačena společná část (všimněte si rozdílů v *id* vrcholů), tenké nepřerušované hrany pocházejí z grafu G, čárkované hrany

z grafu H. (Zde jsme nakreslili jeden z možných výsledků, ostatní se budou lišit tím, který vrchol v grafu H je ve společné části, případně otočením nebo překlopením čtyřúhelníku.)

Všechny operace předpokládají, že dostanou korektní vstup — není tedy například povoleno volat je s *id* neexistujícího vrcholu nebo upravovat grafovou proměnnou, do které jste ještě nepřiřadili, a podobně.

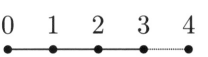
Všechny grafové operace trvají konstantní čas.

Abychom vám usnadnili ladění programů, vytvořili jsme simulátor grafového počítače. Najdete ho od září na webových stránkách olympiády.

Příklad 1: Tvorba cesty

Ukážeme, jak vytvořit *cestu* délky n. To je graf o $n+1$ vrcholech a n hranách, ve kterém je každý vrchol spojen hranou s následujícím. Zajisté bychom cestu mohli vytvářet postupně, například takto:

```
function cesta(n: Integer): Graph;
var
  i, posledni, novy: Integer;
  g: Graph;
begin
  g := EmptyG;
  posledni := AddV(g, 0);
  for i := 1 to n do begin
    novy := AddV(g, 0);
    AddE(g, posledni, novy, undef);
    posledni := novy;
  end;
  cesta := g;
end;
```

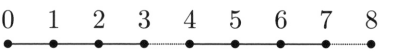


Začínáme s jediným vrcholem (má *id* 1) a pak n-krát přidáme nový vrchol a hranu do něj. (Vrcholům dáváme značky 0, hranám nedefinované váhy, což se bude hodit později.) Celý postup tedy trvá lineárně dlouho a vytvoří cestu začínající ve vrcholu s *id* 1 a končící vrcholem s *id* $n+1$. Nešlo by to rychleji?

Představme si na chvilku, že máme v g již část cesty, řekněme o k vrcholech. Pomocí `Join(g,g,none,none)` vytvoříme nový graf, který obsahuje dvě kopie této cesty (jednu s *id* $1,\ldots,k$, druhou s *id* $k+1,\ldots,2k$). Stačí tedy přidat hranu z k do $k+1$ a máme cestu délky $2k$. Toho využijeme v následujícím (rekurzivním) řešení úlohy:

```
function cesta(n: Integer): Graph;
var
```

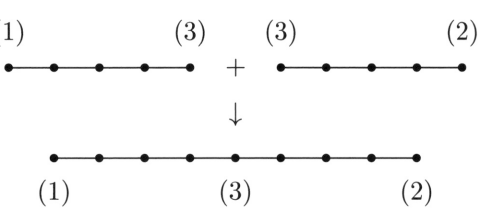
```
  vysledek: Graph;
  pulka: Integer;
begin
  if n = 0 then begin { Cesta délky 0 je snadná }
    vysledek := EmptyG;
    AddV(vysledek, 0);
  end else begin
    { Rekurzivně vytvoříme cestu poloviční délky }
    pulka := (n-1) div 2;
    vysledek := cesta(pulka);
    { Vyrobíme 2 kopie a spojíme je }
    vysledek := Join(vysledek, vysledek, none, none);
    AddE(vysledek, pulka+1, pulka+2, undef);
    { Když polovina nevyšla celočíselně, přidáme ještě hranu }
    if n mod 2 = 0 then begin
      AddV(vysledek, 0);
      AddE(vysledek, n, n+1, undef);
    end
  end;
  cesta := vysledek;
end;
```



Při každém rekurzivním volání se n zmenší alespoň dvakrát, časová složitost tohoto řešení je tedy $\mathcal{O}(\log n)$.

Ukážeme ještě jedno řešení, tentokrát založené na spojování cest za vrchol. Budeme vytvářet cesty, jejichž počáteční vrchol bude mít značku 1, koncový vrchol značku 2 a všechny ostatní vrcholy `undef`. Když chceme dvě cesty spojit do jedné, přeznačíme koncový vrchol první a počáteční vrchol druhé na 3 a zavoláme `Join` s veq = `value_defined`. Tím způsobíme, že se vrcholy označené trojkou ztotožní a vznikne cesta dvojnásobné délky (kdybychom místo `value_defined` použili `value`, ztotožnily by se i vnitřní vrcholy cest, což nechceme). Pak ještě odstraníme pomocnou značku 3 a přepíšeme ji na `undef`. Program tentokrát pro jednoduchost napíšeme pouze pro $n = 2^k$:

```
function cesta(n: Integer): Graph;
var
  g, t1, t2: Graph;
begin
  if n = 1 then begin
    g := EmptyG;
    AddV(g, 1);
    AddV(g, 2);
```



```
    AddE(g, 1, 2, undef);
  end else begin
    t1 := cesta(n div 2);
    t2 := t1;
    ReplaceV(t1, 2, 3);
    ReplaceV(t2, 1, 3);
    g := Join(t1, t2, value_defined, any);
    ReplaceV(g, 3, undef);
  end;
  cesta := g;
end;
```

Časová složitost tohoto řešení je opět logaritmická.

Příklad 2: Obchodní cestující

Všichni známe vykutálené obchodníky. Prodávají kdovíco a nejraději by, kdyby je po prodeji kupující již nikdy nenašel.

Představme si takového obchodníka. Nyní se nachází ve městě (vrcholu) číslo 1. Chce projet celou zemi (graf) po silnicích (hranách) tak, aby navštívil každé město právě jednou a pak se vrátil domů. Navíc při tom chce najezdit co nejméně, takže by celková váha použitých hran měla být co možná nejmenší.

Na obvyklém počítači tento problém neumíme vyřešit v polynomiálním čase, ale pokud máme k dispozici grafový počítač, půjde to velice efektivně.

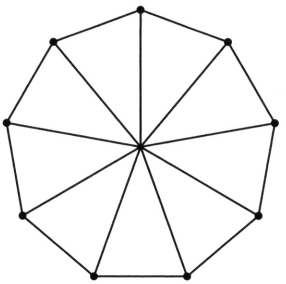
Stačí totiž vyrobit cyklus z n hran a funkcí `Find` nalézt jeho nejlehčí výskyt v grafu popisujícím mapu. Cyklus vytvoříme tak, že podle předchozího příkladu vytvoříme cestu o $n-1$ hranách očíslovanou $1, \ldots, n$ a poté spojíme hranou její první vrchol s posledním. To bude trvat logaritmicky dlouho a funkce `Find` pak konstantně. I program bude jednoduchý:

```
function cestujici(mapa: Graph): Graph;
var trasa: Graph;
begin
  trasa := cesta(CountV(mapa)-1);
  AddE(trasa, 1, CountV(mapa), undef);
  cestujici := Find(mapa, trasa, any, any);
end;
```

Soutěžní úloha.

a) *(5 bodů) Loukoťové kolo* je graf, který vznikne z cyklu o n vrcholech přidáním jednoho vrcholu (osy) spojeného s každým vrcholem cyklu

hranami (loukotěmi). Loukoťové kolo velikosti n má tedy $n + 1$ vrcholů a $2n$ hran.



Napište funkci pro grafový počítač, která pro zadané n takové kolo zkonstruuje. Na obrázku vpravo vidíte příklad takového kola pro $n = 9$.

b) *(5 bodů)* Mějme graf popisující silniční síť: vrcholy jsou města, hrany silnice ohodnocené nezápornými vzdálenostmi v kilometrech. Silnice jsou propojeny pouze ve městech, všechna ostatní křížení jsou mimoúrovňová. Bude nás zajímat, jakou nejmenší vzdálenost musíme ujet, abychom se dostali z města x do města y. Jinými slovy, máme v grafu nalézt cestu mezi x a y, na které bude celkový součet vah hran nejmenší.

Napište funkci pro grafový počítač, která dostane na vstupu graf silniční sítě a identifikátory dvou různých vrcholů x, y, a odpoví vzdáleností mezi těmito vrcholy.

P – II – 1

Vlak

Na nákladním nádraží stál vlak. Sice bez lokomotivy, ale tu měli vzápětí připojit, když tu přišel přednosta stanice, prohlédl si seřazené vagóny a oznámil železničářům nepříjemnou zprávu. Nový předpis mu přikazuje poslat do cílové stanice zprávu, v jakém pořadí budou ve vlaku vagóny řazeny, ale neví, kolikrát vlak na cestě změní směr, takže potřebuje, aby vypadal stejně v obou směrech.

Navíc železničáři nesmí vagóny vyměňovat. Jediné, co mohou, je vagón vyřadit (to jim nařizuje jiný předpis). Dopravce chce zároveň na každém vlaku vydělat co nejvíce, což znamená, že je třeba do jednoho vlaku zařadit co nejvíce vagónů.

Soutěžní úloha. Na vstupu dostanete posloupnost písmen. Každé z těchto písmen představuje typ vagónu. Výstupem vašeho programu bude výpis pozic vagónů (písmen), které musejí být odstraněny, aby výsledný vlak (posloupnost písmen) po jejich odstranění byl stejný při čtení zepředu i zezadu (tedy aby výsledný řetězec písmen byl palindromem).

Pokud existuje více možností, nalezněte a vypište tu, kde je třeba odstranit nejmenší možný počet vagónů. Pokud je takových možností více, můžete vypsat libovolnou z nich.

Formát vstupu: Vstup je tvořen dvěma řádky. První obsahuje celé číslo N ($1 \leqq N \leqq 50\,000$), které udává původní počet vagónů vlaku. Druhý řádek pak obsahuje posloupnost N znaků, které reprezentují jednotlivé typy vagónů.

Formát výstupu: Výstup je tvořen dvěma řádky. První obsahuje jediné číslo K ($0 \leqq K \leqq N - 1$), které udává, kolik vagónů je potřeba z vlaku odstranit, aby vlak vypadal stejně z obou směrů. Druhý řádek pak obsahuje K různých čísel od 1 do N určujících pořadí vagónů, které mají být z vlaku odstraněny. Pokud posloupnost znaků na vstupu je stejná při čtení zepředu i zezadu, pak na první řádek vypište číslo 0 a druhý řádek bude prázdný.

Příklady:

Vstup:	*Výstup:*
6	1
ABCDBA	3

Odstraněním třetího písmene vznikne vlak s vagóny ABDBA. *Jiné optimální řešení je odstranit čtvrté písmeno, kdy vznikne vlak s vagóny* ABCBA.

Vstup:	*Výstup:*
7	2
ABECEDA	2 6

Odstraněním druhého a šestého písmene vznikne vlak s vagóny AECEA.

Vstup:	*Výstup:*
7	4
ABECADA	3 4 6 7

Odstraněním třetího, čtvrtého, šestého a sedmého písmene vznikne vlak s vagóny ABA. *V tomto případě je však optimálních řešení mnohem více.*

P – II – 2

Jabloňový sad

V jednom malém království vyrostl strom se zlatými jablky. Král byl praktický člověk, a tak přikázal zahradníkům, ať z takových jabloní vypěstují celý sad. Ale k jejich nemilému překvapení většina stromů urodila jen obyčejná kyselá jablka. Alchymisté však objevili zvláštní formuli, která říkala, kdy a kam zasadit semínko stromu, aby měl opět zlatá jablka. Prvních N stromů, které takto zasadili, bylo skutečně zlatých, a tak si

král nechal vypracovat seznam, podle něhož má stromy v příštích letech sázet.

Ač stromy rostly překvapivě rychle, ruce nenechavců a zlodějíčků byly ještě rychlejší. Dlouho netrvalo a král začal se stavbou oplocení. Odhad účtu za pletivo byl však zdrcující. Poddaní se totiž rozhodli oplotit čtvrt království, aby všechny stávající i budoucí stromy rostly uvnitř. Krále napadlo, že zřejmě bude lepší oplotit jen nynější stromy a pak podle potřeby rozšiřovat plot i na nové stromy. Aby se ušetřilo, část stávajícího oplocení se rozebere a použije spolu s nově nakoupeným množstvím pletiva. Král si tedy sehnal Vás coby projektanta a netrpělivě očekává vyhotovený rozpočet za pletivo na příštích deset let. Vám je jasné, co musíte spočítat nejdříve: kolik pletiva musíte koupit na počátku a potom se zasazením každého nového stromu. Alchymisté, vidouce Vaše zděšení, Vám však ještě dali jednu radu: Minimální nutná délka plotu se s přidáním nového stromu nikdy nezmenší.

Soutěžní úloha. Na začátku dostanete kartézské souřadnice N bodů (jabloní) v rovině ($[X_1, Y_1]$, ..., $[X_N, Y_N]$) a musíte zjistit nejmenší možný obvod obrazce, který je všechny bude obsahovat (tj. délka oplocení). Postupně obdržíte M dalších bodů zadaných souřadnicemi v rovině. Tyto body budete přidávat ke stávajícím v pořadí od prvního do M-tého (sázejí se nové stromy). Po přidání každého z nich musíte spočítat, o kolik se zvětšil obvod nejmenšího obrazce, který obsahuje všechny původní i dosud přidané body. Výsledkem může být i 0, pokud nově přidaný bod již leží uvnitř obrazce obklopujícího dříve přidané body. Vámi spočítaný údaj tedy odpovídá tomu, kolik pletiva je třeba přikoupit.

M očekávejte jako velké číslo, takže abyste králi mohli dát rozpočet včas, musíte umět potřebnou délku nového pletiva po přidání stromu spočítat rychle. Bude tedy dobrý nápad při přidání každého bodu využít dříve spočítané hodnoty.

Formát vstupu: Na prvním řádku vstupu bude číslo N, které udává počáteční počet jabloní (bodů). Následuje N řádků, kde na i-tém řádku ($1 \leqq i \leqq N$) jsou dvě čísla X_i a Y_i oddělená mezerou (souřadnice již zasazených stromů). Další řádek obsahuje číslo M. Následuje M řádků, kde na i-tém řádku ($1 \leqq i \leqq M$) jsou dvě čísla X_i' a Y_i' oddělená mezerou, která udávají souřadnice nově vysazovaných stromů.

Formát výstupu: Na výstupu vypíšete celkem $M+1$ řádků. Na prvním řádku bude počáteční délka oplocení, tj. obvod nejmenšího obrazce, který obsahuje všechny body $[X_1, Y_1]$, ..., $[X_N, Y_N]$. Následuje M řádků, na

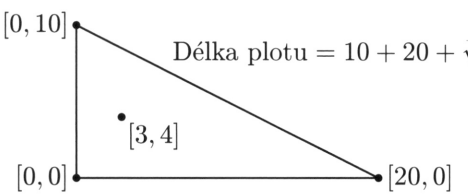
j-tém z nichž ($1 \leqq j \leqq M$) bude uvedeno, o kolik se musí obvod obrazce zvětšit po přidání bodu $[X'_j, Y'_j]$ k předchozím (tj. kolik nového pletiva je zapotřebí). Výsledky vypisujte s přesností na 5 desetinných míst.

Program musí výstup vypisovat průběžně. Pokaždé, když ze vstupu přečte polohu dalšího vysazovaného stromu, musí vydat příslušný řádek výstupu, aniž by čekal, až budou zadány všechny stromy.

Příklad:

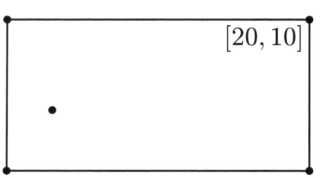
Vstup:	*Výstup:*
`4`	`52.36068`
`0 0`	`7.63932`
`0 10`	`4.14213`
`20 0`	
`3 4`	
`2`	
`20 10`	
`-5 5`	

Počáteční situace:



Délka plotu $= 10 + 20 + \sqrt{10^2 + 20^2} \approx 52{,}360\,68$

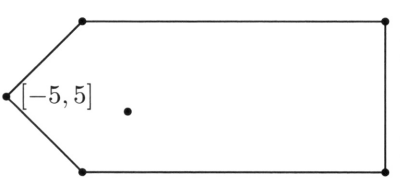
Po přidání bodu $[20, 10]$*:*



Délka plotu $= 10 + 20 + 20 + 10 = 60$

Rozdíl $\approx 7{,}639\,32$

Po přidání bodu $[-5, 5]$*:*



Délka plotu $=$
$= 10 + 20 + 20 + 2\sqrt{5^2 + 5^2} \approx$
$\approx 64{,}142\,14$

Rozdíl $\approx 4{,}142\,14$

P – II – 3

Mažoretky

Kromě stromů se zlatými jablky mají v království ještě jednu zvláštnost. Příchod každého dne slaví v hlavním městě průvodem N mažoretek. V průvodu kráčí mažoretky v jedné řadě za sebou.

Aby se poddaní nenudili, tak v každém dni v roce pochodují mažoretky v jiném pořadí. A protože rok má v království přesně $N! = 1 \cdot 2 \cdot \ldots \cdot N$ dní, tak během roku pochodují v každém svém možném pořadí právě jednou.

Pořadí, v jakém mažoretky pochodují, se určují během roku následovně. Každá mažoretka má své číslo od 1 do N. Jejich pořadí v konkrétním dnu si tedy můžeme představit jako posloupnost N navzájem různých čísel od 1 do N. Pokud $(a_1, \ldots, a_N)$ a $(b_1, \ldots, b_N)$ jsou dvě takové posloupnosti, pak mažoretky pochodují v pořadí $(a_1, \ldots, a_N)$ v jednom roce dříve než v pořadí $(b_1, \ldots, b_N)$, jestliže pro nejmenší index i s $a_i \neq b_i$ platí $a_i < b_i$. Pokud například $N = 4$, pak v pořadí $(3, 1, 2, 4)$ budou mažoretky pochodovat dříve, než v pořadí $(3, 4, 1, 2)$.

Pro $N = 3$, budou mažoretky v jednom roce pochodovat postupně v pořadí $(1, 2, 3)$, $(1, 3, 2)$, $(2, 1, 3)$, $(2, 3, 1)$, $(3, 1, 2)$ a $(3, 2, 1)$.

Soutěžní úloha. Vstup obsahuje dva řádky. Na prvním řádku je jedno celé číslo $N \geqq 2$, které udává počet mažoretek. Druhý řádek obsahuje N navzájem různých celých čísel od 1 do N. Tato čísla určují pořadí mažoretek. Na výstup vypište dva řádky, které obsahují:

a) *(3 body)* pořadí mažoretek v následující den,

b) *(7 bodů)* pořadí mažoretek přesně za půl roku.

Příklad:

Vstup:	*Výstup:*
`5`	`1 4 3 2 5`
`1 4 2 5 3`	`4 1 2 5 3`

P – II – 4

Grafový počítač na kliku

K úloze se vztahuje studijní text z úlohy P–I–4.

Soutěžní úloha. a) Napište funkci pro grafový počítač, která pro zadané číslo n zkonstruuje *úplný graf* K_n. To je graf s n vrcholy, jehož každý

vrchol je spojený hranou s každým. Pokud to neumíte pro obecné n, vyřešte úlohu alespoň pro ta n, která jsou mocninami dvojky.

b) Napište funkci pro grafový počítač, která spočítá *klikovost* zadaného grafu. To je největší možný počet vrcholů zadaného grafu, které jsou spojeny každý s každým. Jinak řečeno, je to největší n, pro které je K_n podgrafem daného grafu.

P – III – 1

Básník Honzík

Honzík Nerudů si spokojeně hověl v lavici a v duchu se procházel po Malé Straně. „Jene, chytej!“, vytrhl ho ze zadumání výkřik spolužačky Božky Němcové a náraz boty do jeho hlavy. Být to kdokoli jiný, jistě by neušel spravedlivému trestu, jenže Boženka byla jeho tajnou láskou už od třetí třídy. Jak rád by ji pozval na procházku podél Vltavy. Leč když se opovážil požádat ji, se smíchem ho odbyla slovy:

Jelikož jsem romantička,
obměkčí mě jen básnička.
Zveršuj tedy žádost svou,
a projdem se nad Vltavou.

Jenže Honzík měl ze slohu vždycky čtyřku a paní učitelce stoupaly vlasy hrůzou, jen se chopil pera. Naštěstí měl dobrého kamaráda Járu Cimrmana. Ten mu pověděl, že psát básně je strašně jednoduché a stačí jen dosáhnout toho, aby se verše co nejvíce rýmovaly (později bude tato teorie publikována pod názvem absolutní rým). To byla sice dobrá rada, ale hledání co nejvíce se rýmujících slov je často velmi obtížné, a tak požádal o pomoc vás.

Soutěžní úloha. Máte dán seznam obsahující N slov skládajících se z malých písmen anglické abecedy. Vaším úkolem je zodpovídat Honzíkovy dotazy, což znamená, že pro jím zadané slovo s máte najít slovo r ze seznamu, které má největší společný koncový úsek se slovem s. Např. slova `Zbynek` a `pelynek` mají společný koncový úsek `ynek` délky 4.

Pokud je v seznamu více takových slov, vyberte z nich lexikograficky nejmenší (lexikografické uspořádání je to, které se používá ve slovnících: nejdříve podle prvního písmena, pak podle druhého atd.; `ch` uvažujeme jako dvě písmenka). Pokud naopak v seznamu není žádné slovo se společným koncovým úsekem délky alespoň 1, vypište `NELZE`.

Protože dotazů může být hodně, snažte se optimalizovat rychlost odpovědi na dotaz i za cenu delšího předzpracování seznamu slov.

Poznámka: Pokud úlohu nedokážete vyřešit efektivně, zkuste popsat alespoň řešení, které z vyhovujících slov vypíše libovolné namísto lexikograficky nejmenšího.

Formát vstupu: Na prvním řádku budou dvě čísla N a K, po kterých následuje N slov seznamu, každé na samostatném řádku. Následuje K slov, $K \leqq 10^4$, opět každé na samostatném řádku, která reprezentují Honzíkovy dotazy. Součet délek všech slov seznamu nepřesáhne 10^6 znaků, tedy speciálně $N \leqq 10^6$. Žádné slovo v seznamu ani žádné z K slov k vyhledání nebude mít víc jak 10^4 znaků.

Formát výstupu: Pro každý z K Honzíkových dotazů vypište na samostatný řádek slovo s maximálním společným koncovým úsekem (případně lexikograficky nejmenší, pokud jich je víc) mezi slovy v seznamu. Pokud žádné takové slovo neexistuje, vypište `NELZE`.

Příklad:

Vstup:	*Výstup:*
`5 3`	`NELZE`
`pluji`	`lituji`
`listuji`	`listuji`
`lituji`	
`nepreji`	
`basnik`	
`bagr`	
`kvituji`	
`dekuji`	

Pro slovo `dekuji` *máme na výběr mezi slovy* `pluji`, `listuji` *a* `lituji`, *která mají stejnou délku společného koncového úseku (určitý společný koncový úsek má i se slovem* `nepreji`, *ale ten má délku pouze dva);* `listuji` *je z nich lexikograficky nejmenší.*

P – III – 2

Úřad

Úřad pro minimalizaci byrokracie zaměstnává několik tisíc úředníků. Ti jsou pro zvýšení efektivity své práce hierarchicky uspořádaní, tj. každý z nich má právě jednoho přímého nadřízeného; jedinou výjimkou je ministr pro minimalizaci byrokracie, který je nejvýše postaveným úřední-

kem a žádného nadřízeného nemá. Každý z úředníků smí vykonávat právě jeden úkon (někteří smí dávat pouze kulatá razítka, někteří pouze hranatá, někteří mají na starosti styk s veřejností, atd.). Výjimkou je opět ministr, o kterém legendy tvrdí, že je schopen vykonávat všechny funkce poskytované úřadem.

Potřebuje-li tedy někdo něco zařídit na úřadě, nejprve si vybere nějakého úředníka, který smí komunikovat s veřejností. Ten už ale nesmí vykonávat žádný jiný úkon, a není mu tedy schopen přímo pomoci. Proto ho pošle za svým nadřízeným. Může-li nadřízený požadovaný úkon provést, učiní tak, jinak zájemce přepošle za svým nadřízeným. A toto se opakuje, dokud zájemce nedorazí k někomu schopnému ho obsloužit.

Teď jsou volby na dohled a voliči si stěžují, že někteří úředníci nic nedělají. Například dává-li úředník a všichni jeho přímí podřízení kulatá razítka, pak se k němu nikdy žádný požadavek nedostane. Obdobně úředník, jehož žádný (ani nepřímý) podřízený nemá na starosti styk s veřejností, nikdy nemusí nic dělat. Potřebovali bychom tedy nalézt všechny takové nezaměstnané úředníky, abychom je mohli povýšit.

Poznamenejme ještě, že ministra nikdy za zbytečného nepovažujeme.

Soutěžní úloha. Úředníci jsou očíslováni přirozenými čísly $1, \ldots, N$, kde úředník číslo 1 je ministr. Pro každého z nich až na ministra máme zadáno číslo jeho nadřízeného, které je vždy menší než číslo úředníka. Pro každého úředníka až na ministra také máme zadáno číslo úkonu, který smí vykonávat. Úkony jsou očíslovány přirozenými čísly $1, \ldots, M$, kde úkon číslo 1 je styk z veřejností. Vypište čísla všech úředníků, kteří nikdy nic nedělají. Úředník číslo k, který smí vykonávat úkon číslo u, něco dělá, jestliže $u = 1$ nebo existuje posloupnost čísel $k = a_1 < a_2 < \ldots < a_t$ taková, že:

- úředník číslo a_i je přímý nadřízený úředníka a_{i+1} pro $1 \leqq i \leqq t - 1$,
- úředník a_t má na starosti styk s veřejností,
- žádný z úředníků a_2, a_3, …, a_{t-1} nevykonává úkon u.

Formát vstupu: Program načte vstupní data ze standardního vstupu. První řádek obsahuje přirozená čísla N a M, udávající počet úředníků a počet typů úkonů. Na následujících $N-1$ řádcích jsou popsáni úředníci kromě ministra. Na i-tém z těchto řádků se nachází dvě čísla n_i a u_i ($1 \leqq n_i \leqq i$, $1 \leqq u_i \leqq M$), kde n_i je číslo přímého nadřízeného úředníka číslo $i + 1$ a u_i je číslo úkonu, který smí vykonávat.

Můžete předpokládat, že počet úkonů M je řádově menší než počet úředníků N.

Formát výstupu: Program vypíše na standardní výstup čísla úředníků, kteří nic nedělají, v libovolném pořadí, oddělená mezerami.

Příklad:

Vstup:	*Výstup:*
10 3	2 3 7
1 2	
2 3	
2 2	
2 2	
1 2	
6 2	
6 1	
4 1	
5 1	

P – III – 3

Grafový počítač v potrubí

K úloze se vztahuje studijní text z úlohy P–I–4.

A právě takový počítač umožnil nový způsob komunikace: potrubní poštu. Taková potrubní pošta sestává z mnoha stanic. Některé dvojice stanic jsou propojeny potrubím, které lze použít k přepravě zpráv v obou směrech. Stanice jsou samozřejmě schopné zprávy předávat dál, takže zásilky obvykle putují do cílové stanice několika na sebe navazujícími rourami.

Potrubí bylo postaveno a ještě než došlo k vyřízení všech povolení, nahromadilo se mnoho zpráv, které je třeba doručit. A protože je to systém nový, rozhodli se poštmistři, že začnou posílat od těch nejkratších zpráv, aby zjistili, jestli se v potrubí nezasekávají.

Navíc se po dobu vyřizování formalit v potrubí usadily myši. Myši samozřejmě každé procházející psaní hned zhltnou, proto je potřeba poslat potrubím napřed kočku. Protože však kočka je mnohem těžší než psaní, je také nákladnější ji potrubím profouknout. Proto bylo rozhodnuto, že budou vyčištěny jen některé roury, a to tak, aby jejich celková délka byla co nejmenší a přitom bylo možno poslat psaní z libovolné stanice do libovolné jiné.

Soutěžní úloha. a) *(3 body)* Napište funkci pro grafový počítač, která seřadí zprávy podle délky. Vstupem funkce bude pole celých čísel — délek

zpráv. Úkolem je toto pole setřídit od nejmenšího po největší. Můžete při tom využít toho, že délky zpráv se vejdou do typu `Value` grafového počítače.

Řešení s časovou složitostí $\mathcal{O}(n \log n)$ může získat nejvýše 1 bod, pomalejší řešení nedostanou žádné body.

b) *(7 bodů)* Dostanete na vstupu popis potrubí jako souvislý ohodnocený graf (stanice jsou vrcholy, trubky jsou hrany a jejich váhy odpovídají délkám trubek). Vraťte podgraf obsahující právě ty hrany, které mají být vyčištěny. Jinými slovy podgraf, ve kterém vede mezi každou dvojicí vrcholů cesta a který má ze všech takových grafů nejmenší možný součet vah hran.

Pokud vám to pomůže, můžete předpokládat, že neexistuje žádná dvojice stejně dlouhých rour.

I u této podúlohy bude při hodnocení kladen důraz na to, zda je vaše řešení rychlejší než řešení, která se dají naprogramovat na klasickém počítači.

P – III – 4

Asfaltistán

Program: `asfalt.pas` / `asfalt.c` / `asfalt.cpp`
Vstup: `asfalt.in`
Výstup: `asfalt.out`

V Asfaltistánu zjistili, že mají jedno velké území dosud nedotčené asfaltem. Rozhodli se to napravit tak, že napříč územím postaví silnici. Nicméně ministr financí si klade podmínky. Silnice musí být sjízdná pro auta a musí být postavena co nejlevněji. Je totiž krize a musí se šetřit.

Projektanti si rozdělili území na $R \times S$ čtverců o straně 1 km a pro zjednodušení si pro každý čtverec spočítali jeho průměrnou nadmořskou výšku. Zjistili, že silnice mezi dvěma sousedními čtverci je sjízdná, pokud je rozdíl jejich nadmořských výšek maximálně 1 obří sáh.

Navíc můžou stavět mosty a tunely — mostem je možno propojit dva čtverce ve stejném sloupci nebo řádku, pokud je jejich nadmořská výška shodná a nadmořská výška všech čtverců mezi nimi je nižší. Tunelem je také možno propojit dva čtverce o stejné nadmořské výšce ve stejném sloupci nebo řádku, jen nadmořská výška všech čtverců mezi nimi musí být vyšší.

Vaším úkolem je najít nejlevnější silnici mezi políčkem $(0,0)$ a políčkem $(R-1, S-1)$. Silnice musí být na políčkách $(0,0)$ a $(R-1, S-1)$ na povrchu, tedy ne na mostě ani v tunelu.

Poznamenejme, že na jednom políčku může jeden most/tunel končit a současně další začínat. Také si všimněte, že je teoreticky možné, že se na optimální cestě budou dva mosty nebo tunely křižovat. To je povolené, šikovní asfaltistánští inženýři to dokáží vyřešit tak, aby se auta nesrážela.

Formát vstupu: Na vstupu dostanete na prvním řádku celá čísla R, S, c_S, c_M a c_T. Čísla c_S, c_M a c_T udávají ceny za postavení jednoho políčka silnice, mostu a tunelu (v asfaltových dolarech).

Cena za políčko mostu nebo tunelu se počítá jen za políčka *uvnitř*, čili začátek a konec tunelu/mostu se nepočítá.

Na každém z dalších R řádků dostanete S čísel $h_{i,j}$ oddělených mezerou — nadmořské výšky jednotlivých políček v obřích sázích. Na i-tém řádku v j-tém sloupci je uvedeno číslo $h_{i,j}$.

Formát výstupu: Na první řádek výstupu vypište celkovou cenu silnice v asfaltových dolarech. Na následující řádky vypište trasu silnice; na každý řádek dvojici (i_k, j_k) udávající políčko, přes které silnice prochází. Políčka, která se nachází pod mostem nebo nad tunelem, vynechte. První souřadnice vždy udává řádek a druhá sloupec. Pokud řešení neexistuje, vypište na jediném řádku výstupu slovo `NEEXISTUJE`.

Všimněte si, že celková cena může být značně velké číslo, které se do 32bitové proměnné nemusí vejít.

Velikost vstupu: Ve všech vstupech použitých při testování platí $1 \leqq R, S \leqq 1\,000$, $1 \leqq c_S, c_M, c_T \leqq 1\,000\,000$ a $0 \leqq h_{i,j} \leqq 1\,000\,000$.

Ve vstupech za alespoň 5 bodů mimo to platí $1 \leqq R, S \leqq 50$.

Ve vstupech za alespoň 5 bodů také platí, že alespoň jedna z nejlevnějších silnic neobsahuje žádný tunel ani most. Tyto vstupy nemusejí být různé od těch uvedených v předchozím odstavci.

Příklady:

Vstup:	*Výstup:*
2 7 1 1 20 5 1 1 5 5 5 5 5 5 5 5 9 9 5	8 0 0 0 3 0 4 0 5 0 6 1 6

Vstup:	*Výstup:*
2 7 1 20 1 5 1 1 5 5 5 5 5 5 5 5 9 9 5	8 0 0 1 0 1 1 1 2 1 3 1 6

Vstup:	*Výstup:*	*Vstup:*	*Výstup:*
2 7 1 20 20	10	3 3 1 1 1	NEEXISTUJE
5 1 1 5 5 5 5	0 0	42 32 32	
5 5 5 5 9 9 5	1 0	32 22 12	
	1 1	22 12 2	
	1 2		
	1 3		
	0 3		
	0 4		
	0 5		
	0 6		
	1 6		

P – III – 5

Básník Honzík II

Program: `basnik.pas` / `basnik.c` / `basnik.cpp`
Vstup: `basnik.in`
Výstup: `basnik.out`

Honzík poté, co jste mu včera pomohli pozvat Boženku na procházku, zjistil, že v sobě ukrývá velký básnický talent, a chtěl by po vás poradit ještě jednou. Rozhodl se totiž, že složí nejkrásnější báseň všech dob. Sousedovic Kája Máchů mu poradil, že v nejkrásnějších básních má každá sloka dva verše. Pak od svého kamaráda Jardy Seifertíka zjistil, že každá správná básnička má mít přesně N slok. Nakonec se od Elišky Hezkohorské dozvěděl, že tyto básně mají takzvaný cyklický sdružený rým. To znamená, že mají schéma *ab bc cd de ef* ... *yz za*, tedy že druhý verš každé sloky se rýmuje s prvním veršem té následující; výjimku tvoří druhý verš poslední sloky, který se rýmuje s prvním veršem první sloky.

Honzík má skutečně bohatou fantazii a navíc i váš včerejší program, takže pro něj nebyl problém vymyslet jednotlivé sloky i jejich pozici v básni. Co ale čert nechtěl, od některých slok vymyslel několik různých variant, které sice vyjadřují stejnou myšlenku, ale obsahují jiné dvojice rýmů.

A právě s výběrem jednotlivých slok do výsledné básně by chtěl pomoci. Pro každou z N výsledných slok dostanete S_i variant, které Honzík vymyslel, a vaším úkolem je sestavit básničku tak, aby měla cyklický sdružený rým.

Abyste si nemuseli lámat hlavu s tím, které verše se rýmují a které ne, očísloval Honzík všechny možné rýmy přirozenými čísly. Každou sloku pak popsal dvojicí (x, y) obsahující čísla rýmů v obou verších. Za slokou (x, y) se tedy může vyskytovat sloka (a, b) právě tehdy, když $y = a$.

Soutěžní úloha. Napište program, který pro každou sloku číslo i ($1 \leqq \leqq i \leqq N$) dostane S_i možných variant a z nich sestaví básničku, která splňuje požadované schéma rýmů.

Formát vstupu: První řádek vstupního souboru obsahuje přirozené číslo N, které udává počet slok básničky. Následuje N skupin řádků, přičemž skupina i začíná řádkem obsahujícím číslo S_i následované S_i řádky, každý s dvěma čísly A a B. Číslo A udává rým, na který končí první verš varianty, a číslo B udává rým, na který končí druhý verš.

Formát výstupu: Program vypíše do výstupního souboru buď řádek s řetězcem `NEEXISTUJE`, pokud básničku sestavit nelze, nebo N řádek, přičemž i-tý řádek bude obsahovat číslo k ($1 \leqq k \leqq S_i$), které určuje vybranou variantu pro sloku číslo i. Pokud existuje více řešení, vypište libovolné z nich.

Velikost vstupu: Všechny vstupy použité při testování mají $N \leqq 1\,000$, $1 \leqq S_i \leqq 1\,000$ pro všechna i a $1 \leqq A, B \leqq 10^9$.

Ve vstupech celkem ohodnocených 5 body mimo to platí $N \leqq 10$ a $S_i \leqq 10$.

Příklady:

Vstup:	*Výstup:*
5 1 1 1 2 1 5 1 6 2 5 2 6 3 1 3 8 3 10 2 8 1 4 2	1 2 2 1 2

Jednotlivé sloky básničky budou mít rýmy 11 16 63 38 81.

Vstup:	*Výstup:*
2 1 2 1 1 1 3	NEEXISTUJE