Korešpondenčný seminár z programovania
XVIII. ročník, 2000/2001
Katedra vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

KSP finančne podporuje nadácia Open Society Fund Bratislava a IUVENTA – zariadenie pre voľný čas detí, mládeže a dospelých

Vzorové riešenia 3. kola

Milí naši riešitelia,
konečne sme sa dočkali! My sme sa dočkali peknej, všade rozvoniavajúcej jari. A vy, pre zmenu, čerstvých[1] vzorových riešení.

Radi by sme vás upornili, že **18. mája o 13:45** sa uskutoční už 3. ročník internetovej sútaže **Internet Problem Solving Contest - IPSC**. Ide o online programátorskú sútaž trojčlenných teamov trvajúcu 5 hodín. Všetka komunikácia prebiaha výhradne v angličtine prostredníctvom internetu. Na sútaž sa treba vopred registrovať. Všetky relevantné informácie vrátane minuloročných príkladov, ilustračných príkladov, pravidiel, technických informácií a registračných formulárov nájdete na **ipsc.ksp.sk**. Preto neváhajte a zapojte sa!

Kto nežere, neni chlap.

KSPáci

JABBERWOCKY
'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves,
And the mome raths outgrabe.

opravoval si MišoF.
(max. 15 bodov)

1. O zúfalom programátorovi

Nuž, vcelku ste ma potešili, dokonca aj riešenie podobné tomu, ktoré som mal ja na mysli (len sa mi ho nechcelo písať) niektorí poslali. A ja vám ho teraz aj tak musím písať ako vzorové riešenie... No, čo už narobím? Hor sa na to!

Všimnime si najskôr pesničky, ktoré sa ani raz nepohnú. Tieto pesničky už musia byť utriedené, lebo na konci budú v takom istom poradí, ako sú teraz. Preto pesničky, ktorými nehýbeme, tvoria vybranú rastúcu podpostupnosť. Ak označíme dĺžku najdlhšej vybranej rastúcej podpostupnosti D, vieme z toho, že nepohnúť sa môže najviac D, alebo inými slovami určite musíme pohnúť aspoň $N - D$ pesničkami.

Majme teraz nejakú postupnosť (permutáciu čísel 1 až N) s najdlhšou vybranou rastúcou podpostupnosťou dĺžky $D < N$. Zoberme ľubovoľný z prvkov, ktoré do nej nepatria a presuňme ho medzi tie dva prvky podpostupnosti, medzi ktorými leží (prípadne na začiatok, ak je menší od najmenšieho z nich, resp. na koniec v opačnom prípade). Týmto dostaneme novú postupnosť, ktorej najdlhšia vybraná rastúca podpostupnosť má určite dĺžku aspoň $D + 1$. To preto, že jednou z vybraných rastúcich podpostupností je tá z pôvodnej postupnosti s presunutým prvkom navyše. Preto ľubovoľnú postupnosť, ktorej najdlhšia vybraná rastúca podpostupnosť má dĺžku D, vieme určite utriediť na najviac $N - D$ presunov.[2]

[1] Pozor, aby ste sa nepopálili.

[2] Toto tvrdenie sa tiež dalo ukázať cez to, že každou pesničkou sa nám oplatí pohnúť najviac raz, ale toto sa mi zdalo elegantnejšie.

Z toho nám vyplýva potešiteľný záver – hľadaný počet presunov je $N - D$. Ako teraz nájsť dĺžku najdlhšej vybranej rastúcej podpostupnosti? Jednoduché riešenie bežiace v čase $O(N^2)$ využíva metódu dynamického programovania. Keď pre každý z prvých i prvkov vieme dĺžku najdlhšej vybranej rastúcej podpostupnosti, ktorá ním končí, pre $i+1$. prvok ju určíme ľahko – tento môžeme pridať na koniec ľubovoľnej postupnosti, ktorá končí skorším prvkom menším od tohto. V najhoršom prípade (ak boli všetky pred ním väčšie) bude tento prvok tvoriť celú postupnosť. Takže dĺžka najdlhšej podpostupnosti končiacej $i + 1$. prvkom je

$$\text{dlzka[i + 1]} = 1 \ + \ \max\Big\{\{0\} \cup \{\text{dlzka[j]} \mid 1 \leq j \leq i \wedge \text{prvok[j]} < \text{prvok[i + 1]}\}\Big\}$$

Túto hodnotu ľahko spočítame jedným cyklom.

A čo takto lepšie riešenie? To naše bude bežať v čase $O(N.\log D)$ a bude fungovať nasledovne:

Budeme postupne spracovávať prvky postupnosti. Pre každú hodnotu $i \in \{1, \ldots, D\}$ si budeme pamätať najmenšiu hodnotu prvku, akým doteraz končila nejaká vybraná rastúca podpostupnosť dĺžky i. Kvôli ľahšej implementácii si pre nulu budeme pamätať nulu, čo vlastne znamená len toľko, že do prázdnej postupnosti môžeme pridať ľubovoľný prvok. Teda napríklad po spracovaní postupnosti $1, 4, 5, 2$ si budeme pamätať hodnoty $1, 2, 5$. (Napr. podpostupnosti dĺžky 2 sú (1,4),(1,5),(1,2) a (4,5), najmenší z koncov je 2.) O týchto hodnotách vieme povedať, že budú rastúce, lebo keď zoberieme podpostupnosť dĺžky $i + 1$, ktorá má zo všetkých podpostupností dĺžky $i + 1$ najmenší posledný prvok, jej prvých i prvkov tvorí i prvkovú podpostupnosť, ktorej posledný prvok je menší ako posledný prvok ľubovoľnej $(i + 1)$-prvkovej.

Ako sa nám tieto hodnoty zmenia, keď pridáme na koniec postupnosti ďalší prvok s hodnotou x? Nech $j \geq 0$ je najväčšie také, že existuje postupnosť dĺžky j, končiaca menším prvkom, ale už neexistuje žiadna dlhšia (t.j. neexistuje $(j + 1)$-prvková). Hodnoty pre 1 až j sa nám nezmenia, lebo keby sme aj tento prvok pridali na koniec nejakej postupnosti, nedostaneme podpostupnosť s menším koncom, ako sme už našli. Toto vyplýva z faktu, že pamätané hodnoty sú rastúce a j-ta je menšia ako x. Taktiež sa nám nezmenia hodnoty od $j + 2$ ďalej. Aby sme totiž dostali aspoň $j + 2$-prvkovú podpostupnosť, končiacu týmto prvkom, museli by sme ho pridať na koniec nejakej aspoň $(j + 1)$-prvkovej. Ale každá aspoň $(j + 1)$-prvková podpostupnosť končí väčším (vo všeobecnosti, keď môžu byť aj rovnaké hodnoty, väčším alebo rovným) prvkom ako x. Preto jediná hodnota, ktorá sa zmení, bude $(j + 1)$. a zmení sa na x. Doteraz totiž každá $(j + 1)$-prvková podpostupnosť končila číslom väčším ako x. Keď ale pridáme x na koniec podpostupnosti dĺžky j s najmenším posledným prvkom, ktorý je menší ako x, dostaneme podpostupnosť dĺžky $j + 1$, končiacu prvkom x.

Náš algoritmus teda len pre každý prvok nájde hodnotu j a upraví príslušnú zapamätanú hodnotu. Keďže zapamätané hodnoty sú rastúce, hodnotu j môžeme nájsť binárnym vyhľadávaním, čo pri dobrej implementácii vedie k riešeniu s časovou zložitosťou $O(N.\log D)$, ako som sľúbil.

Tradične na záver, ako som vaše riešenia hodnotil?

'Beware the Jabberwock, my son!
The jaws that bite, the claws that catch!
Beware the Fubjub bird, and shun
The frumious Bandersnatch!'
He took his vorpal sword in hand:
Long time the manxome foe he sought–
So rested he by the Tumtum tree,
And stood awhile in thought.

- Vzorové riešenia, ktoré potrebovali čas $O(N.\log D)$ – 15 bodov

- Takmer vzorové riešenie, ktoré potrebovalo čas $O(N.\log N)$ – 14 bodov
- Riešenia, ktoré potrebovali čas $O(N^2)$ – max. 11 bodov
- Nesprávne riešenia – max. 2 body podľa toho, či sa myšlienka aspoň podobala na riešenie a mala hlavu a pätu.

Mnohí ste vo svojich riešeniach nepovažovali za potrebné zdôvodniť, prečo vlastne hľadáte najdlhšiu vybranú rastúcu podpostupnosť, mnohí zdôvodnili len polovicu (buď neukázali, že menej nestačí, alebo ukázali, že toľko stačí, ale nie či nejde menej). Preto som bol tentokrát dosť prísny a za popis sa dali stratiť až dva body. Tiež samozrejme šli body dole za chyby v programe.

Listing programu:

```
program Zufaly_programator_riesi_svoj_problem;
const MaxN = 1000;

var koniec : array[0..MaxN] of integer;
    N,D,i,j,x : integer;

function bsearch(l,r,co : integer) : integer;
{udrzuje invariant koniec[l]<co<=koniec[r]}
begin
   if l+1=r then begin bsearch:=l; exit; end;
   if koniec[(l+r) div 2]<co then
      bsearch:=bsearch((l+r) div 2,r,co)
   else
{ And, as in uffish thought he stood,
     The Jabberwock, with eyes of flame,
  Came whiffling through the tulgey wood,
     And burbled as it came!  }

    bsearch:=bsearch(l,(l+r) div 2,co);
end;

begin
   write('N:'); readln(N);
   write('A[1]:'); readln(x);
   D:=1;
   koniec[0]:=0; koniec[1]:=x;
   for i:=2 to N do begin
      write('A[',i,']:'); readln(x);
      j:=bsearch(0,D+1,x);
      koniec[j+1]:=x;
      if j+1>D then D:=j+1;
   end;
   writeln('Treba presunov: ',N-D);
end.
```

2. O maškrtnom Dávidkovi

opravoval Goober
(max. 15 bodov)

Tento príklad zjavne patril medzi tie ľahšie, pretože väčšina vašich riešení bola správna. Ako je teda možné, že nikto nedostal plný počet bodov? Nuž... nikto z vás nenašiel naj-

lepšie (lineárne) riešenie, ktoré by bolo hodné pätnástich bodov. Riešenia pracujúce v čase $O(N.\log N)$ dostali 11 bodov, kvadratické 8 a nesprávne nanajvýš 3. Ako vždy, chýbajúci popis bol primerane odmenený.

One, two! One, two! And through and through
The vorpal blade went snicker-snack!
He left it dead, and with its head
He went galumphing back.

Ako sa to teda dalo robiť? Základom úspechu bolo zistenie, že maškrty treba vyberať podľa jednotkovej ceny (cena za kilogram). Ak vždy vyberieme mašrtu najvyššej jednotkovej ceny a z nej zoberieme toľko, koľko sa nám ešte do tašky zmestí, určite získame najdrahší možný nákup. Vždy je totiž zaplniť tašku drahšou vecou rovnakého množstva ako vecou lacnejšou. Najjednoduchšie riešenie bolo priamou implementáciou uvedeného postupu – vždy vyhľadáme najdrahšiu vec, vezmeme jej čo najviac a opakujeme, kým je v taške miesto. Vyhľadávanie sa dalo ešte urýchliť použitím haldy alebo úplným utriedením maškŕt (najlepšie niektorým rýchlym algoritmom) podľa jednotkovej ceny. Oba postupy viedli k riešeniam so zložitosťou $O(N.\log N)$.

A ako to robí vzorové riešenie? Rozdeľme maškrty na „drahé“ a „lacné“ (s vyššou a nižšou jednotkovou cenou) tak, aby boli obe skupiny čo najpodobnejšie (čo do počtu druhov maškŕt, nie čo do hmotnosti). Ak sa nám všetky drahé maškrty zmestia do tašky, tak ich môžeme vziať všetky a tašku môžeme doplníme nejakými lacnými. Ak sa ani drahé všetky nezmestia, nemá zmysel o lacných vôbec uvažovať. V oboch prípadoch nám zostane asi polovičná sada maškŕt, ktorými sa musíme zaoberať. So zostávajúcimi maškrtami tento postup opakujeme. Takto môžeme pokračovať, kým nám nezostane posledná maškrta. Tú môžeme v prípade potreby rozdeliť, aby sme vyplnili celú tašku. Prvýkrát teda spracovávame N maškŕt, potom $N/2$, $N/4$, ... Keďže každé spracovanie vieme spraviť v lineárnom čase, celkový čas bude $O(N + N/2 + N/4 + \ldots) = O(2.N) = O(N)$. Pamäť bude, pravdaže, tiež lineárna.

Rozdeľovanie Existuje algoritmus, ktorý umožňuje rozdeliť pole presne podľa našich požiadaviek vždy v lineárnom čase. Ten využíva postup zvaný „Medián z piatich“, ktorý slúži na nájdenie najvhodnejšieho rozdeľovacieho prvku. Z pedagogických príčin tento algoritmus neuvádzame. Radšej zmiernime podmienky kladené na rozdelenie a použijeme postup, ktorý je lineárny „iba“ v priemernom prípade.

‘And hast thou slain the Jabberwock?
Come to my arms, my beamish boy!
O frabjous day! Callooh! Callay!’
He chortled in his joy.

Pri ňom rozdelíme pole vždy na dve časti podľa náhodne zvoleného prvku. „Drahšiu“ časť hneď skúsime dať do tašky, ak sa tam nezmestí, začneme deliť ju, inak ju vezmeme celú a začneme deliť „lacnejšie“ maškrty. Dá sa ukázať, že v priemernom prípade bude celková zložitosť tohoto postupu lineárna.

Listing programu:

```
#define MAX 1000

typedef struct maskrta { float zober,cena,jc,hmota; } MASKRTA;

float m,total;
int n,i,c;    /* n=pocet maskrt */
MASKRTA masy[MAX];  /* masKRTy */
int cisla[MAX];    /* cisla maskrt pri rozdelovani */
```

```
// 'And hast thou slain the Jabberwock?
//    Come to my arms, my beamish boy!
// O frabjous day! Callooh! Callay!'
//    He chortled in his joy.

float min(float a, float b) { return (a<=b)?a:b; }
float hmota(i) { return masy[cisla[i]].hmota; }
float cena(i) { return masy[cisla[i]].cena; }

int Skus_zobrat(int l, int r)
 {
  float sum=0;
  int i;

  for (i=l;i<=r;i++) sum+=hmota(i);
  if (sum>m) return 0;

  m-=sum; for (i=l;i<=r;i++) masy[cisla[i]].zober=hmota(i);
  return 1;
 }

void rozdeluj(int b, int e)
 {
  int i,j;
  float x;

  if (b>e) return;

  /* Ak nam zostala len jedna maskrta, skusme ju rozkrajat */
  if (b==e) masy[cisla[b]].zober=min(hmota(b),m);

  /* Inak rozdelme masy podla niektorej z nich */
  else
   {
    i=b; j=e; x=masy[(i+j)>>1].jc;
    do
     {
      while (masy[cisla[i]].jc>x) i++;
      while (x>masy[cisla[j]].jc) j--;
      if (i<=j)
       {
        c=cisla[i]; cisla[i++]=cisla[j]; cisla[j--]=c;
       }
     } while (i<=j);
    if (j<=b) j=b+1;

   /* Skusime zobrat vsetky drahe veci */
    if (Skus_zobrat(b,j-1)) rozdeluj(j,e);
    else rozdeluj(b,j-1);
   }
 }
```

```
void main()
 {
  scanf("%d %f",&n,&m);
  for (i=0;i<n;i++)
   {
    scanf("%f %f",&masy[i].hmota,&masy[i].cena);
    masy[i].jc=masy[i].cena/masy[i].hmota;
    masy[i].zober=0;
    cisla[i]=i;
   }
  rozdeluj(0,n-1);
  total=0;
  for (i=0;i<n;i++)
   {
    printf("%f ",masy[i].zober);
    total+=masy[i].zober/masy[i].hmota*masy[i].cena;
   }
  printf("\n%f\n",total);
 }
```

'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves,
And the mome raths outgrabe.

Opravoval JanoS
(max. 15 bodov)

3. O Santovi, Bantovi a parcelach II

Tento príklad bol relatívne ľahký a skoro všetci, ktorí ste sa ho odhodlali riešiť, ste ho mali dobre. Najprv zopár detailov k zadaniu. Keby sme od vás nechceli vrcholy daného mnohouholníka, bolo by správnym riešením vypísanie ľubovľného čísla, napríklad 7. Totiž keby sme urobili lomenú čiaru po všetkých vrcholoch a potom sa vrátili po podobnej čiare len o chlp posunutej, obsah by bol zanedbateľne malý. No a potom už len stačí pridať nejaký obdĺžnžnik s obsahom 7. Ak však chceme aj vypísať vrcholy, úloha nie je až takou triviálnou.

In winter, when the fields are white,
I sing this song for your delight

Riešenia som bodoval podľa ich časovovej efektivity a podľa toho, koľko vlastných vrcholov si vytvárali. Dalo sa to samozrejme aj bez nových vrcholov, takže za konštantný počet vrcholov navyše bol -1 bod a za lineárny počet -2 body. Programy s časovou zložitosťou $O(N \log N)$ dostali plných 15 bodov, za $O(N^{3/2})$ som udelil 14 bodov a za riešenia so zložitosťou $O(N^2)$ som dal 12 bodov. Ďalšie body ste mohli stratiť klasicky za popis programu, úhľadnosť a drobné chyby.

Tak a teraz hor' sa na vzorák. Idea je nasledovná: Vezmime si jeden vrchol x_1 a ostatné si utrieďme podľa uhla, aký s osou x zviera vektor (x_1, x_i). Ak majú dva vektory rovnaký uhol, utriedime ich podľa vzdialenosti od tohoto bodu. Teraz si môžete všimnúť, že mnohouholník tvorený bodmi $x_0, x_1, \ldots, x_n$ sa dá rozbiť na $n-2$ trojuholníkov tvaru

In spring, when woods are getting green,
I'll try and tell you what I mean:

x_0, x_i, x_{i+1}. Triedenie je dôležité na to, aby sme sa uistili, že z pohľadu bodu x_1 vidno každý iný vrchol. Obsah trojuholníka vyrátame napríklad vektorovým súčinom a výpočet je hotový.

Listing programu:

```
Program Zlatokopiiiiiiiiiiiiiii;
Var N,I:Integer;
    S:Real;                    {TG-Uhol  D-distance}
    A:Array[0..100] Of Record X,Y,TG,D:Real; End;

    { In summer, when the days are long, }
    { Perhaps you'll understand the song: }

Function Uhol(X,Y:Real):Real;{Uhol vektora X,Y}
Var U:Real;
Begin
   If x=0 Then If Y>0 Then Uhol:=Pi/2 Else Uhol:=-Pi/2
   Else Begin
      U:=Arctan(Y/X);      {Neda spravny vysledok v polovici pripadov}
      If X<0 Then U:=Pi+U; {Opravime nespravnu polrovinu}
      Uhol:=U;
   End;
End;

Procedure QSort(L,R:Integer); {Triedenie v N log N}
Var I,J:Integer;
    X,Y:Real;
Begin
  I := L;  X := A[(l+r) DIV 2].tg;  J := R;  Y := A[(l+r) Div 2].D;
  Repeat
    While (a[i].tg < x) Or ((A[I].D<y) and (A[i].tg=x)) do i := i + 1;
    While (x < a[j].tg) Or ((A[J].D>y) and (A[J].tg=x)) do j := j - 1;
    If I <= J Then Begin
      a[0] := a[i]; a[i] := a[j]; a[j] := a[0];
      I := I + 1; J := J - 1;
    End;
  Until i > j;
  If l < j then QSort(l, j);
  { In autumn, when the leaves are brown, }
  { Take pen and ink, and write it down. }
  If i < r then QSort(i, r);
End;

Begin
   Readln(N);
   For I:=1 To N Do Begin
      Readln(A[I].x,A[I].y);
      If I>1 Then Begin   {Vyratame uhly a stvorce vzdialenosti}
         A[I].Tg:=Uhol(A[I].x-A[1].x,A[I].y-A[1].y);
         A[I].D:=Sqr(A[I].x-A[1].x)+Sqr(A[I].y-A[1].y);
      End;
```

```
  End;
  QSort(2,N); {Utriedime podla uhla}
  S:=0;                    {Vektorovy sucin}
  For I:=2 TO N-1 Do
    S:=S+Abs((A[I].x-A[1].x)*(A[I+1].y-A[1].y)-
             (A[I].y-A[1].y)*(A[I+1].x-A[1].x))/2;
  Writeln('Obsah je ',S:0:2);
  Writeln('Body su:');
  For I:=1 To N Do Writeln('(',A[I].x:0:2,',',A[I].y:0:2,')');
End.
```

opravoval Dávidko
(max. 15 bodov)

4. O futbalovom družstve

Prišla len pomerne malá kôpka riešení. Z nich bolo niekoľko nefunkčných - maximálne 1 bod. Niektorí z vás sa neunúvali písať program - maximálne 5 bodov. Zvyšným riešeniam som udeľoval body podľa zložitosti:

- Exponeciálne a horšie riešenia - maximálne 7 bodov.
- Riešenia $O(N^4)$ a podobné - maximálne 10 bodov.

I sent a message to the fish:
I told them „This is what I wish.“

- Riešenia $O(N^3)$ a podobné - maximálne 12 bodov.
- Riešenia $O(N^2)$ a podobné - maximálne 15 bodov.

Vzorové riešenie: (na základe riešení Mareka Tesaře a Tomáša Záthureckého)

Majme teda naraz všetkých N futbalistov. Postupne po kúsku budeme hľadať najlepšie možné zostavy družstiev (t.j. zostavy s maximálnym súčtom kvalít). Začneme s najlepším družstvom pozostávajúcim z 1 útočníka, 0 poliarov, 0 obrancov (skrátene **optimálne** $(1,0,0)$ – **družstvo**), potom rátame optimálne $(2,0,0)$ – družstvo, atď. ... až optimálne $(A,0,0)$ – družstvo. Podobne pokračujeme s optimálnym $(A,1,0)$ – družstvom atď. ... až optimálnym $(A,B,0)$ – družstvom. A nakoniec z neho vypočítame optimálne $(A,B,1)$ – družstvo atď. ... až optimálne (A,B,C) – družstvo.

The little fishes of the sea,
They sent an answer back to me.

1. fáza, t.j. počítanie optimálneho $(A,0,0)$ – družstva, je ľahká. Najlepšie družstvo skladajúce sa zo samých útočníkov sa skladá z prvých A najlepších útočníkov.

2. fáza: Počítajme optimálne $(A,i+1,0)$ – družstvo, ak už poznáme optimálne $(A,i,0)$ – družstvo $(0 \leq i < B)$. Teraz to chce zopár netriviálnych úvah.

Zostavu družstva si predstavme ako 4 množiny M_A, M_B, M_C, M_D. Po rade sú to množina útočníkov, množina poliarov, množina obrancov a množina nehrajúcich. Optimálne $(A,i+1,0)$ – družstvo sa iste dá z optimálneho $(A,i,0)$ – družstva dostať tak, že budeme po jednom presúvať jednotlivých hráčov medzi množinami (zatiaľ nikto nič nehovorí o tom, ako budú také presuny vyzerať). Presúvaním hráčov medzi množinami sa mení kvalita družstva –

The little fishes' answer was
„We cannot do it, Sir, because –“

– klesá alebo stúpa. Výhodnosťou presunu budeme volať rozdiel kvality družstva po presune a pred presunom (je to vlastne rozdiel kvalít presúvaného hráča). Zrejme nemá význam hýbať niektorým hráčom viac ako raz, preto tak robiť nebudeme.

Pri takomto presúvaní nemôže existovať k-tica hráčov $(2 \leq k \leq 4)$, ktorými sme hýbali a navyše takých, že pred aj po presune ležia v rôznych množinách. Výhodnosť ich presunov

spolu by bola nutne záporná, keďže naše $(A, i, 0)$ – družstvo je optimálne. Preto by sme nimi nemuseli pohnúť, a mali by sme zaručene lepšie družstvo, čo je spor ako hrom. [3]

Po hlbšej úvahe a ešte dlhšom kreslení príde človek na to, že postupností presunov, ktoré zmenia naše optimálne $(A, i, 0)$ – družstvo na optimálne $(A, i + 1, 0)$ – družstvo splňujúcich vyššie uvedenú podmienku nebude nijako hrozne veľa: Buď presunieme niekoho z M_D do M_B, alebo presunieme niekoho z M_A do M_B a následne presunieme niekoho z M_D do M_A.

I sent to them again to say
„It will be better to obey.“

Ak si navyše uvedomíme, že z týchto rôznych postupností presunov máme hľadať najvýhodnejšiu, dostaneme len dva možné presuny: Buď presunieme najlepšieho poliara z M_D do M_B, alebo presunieme najvýhodnejšieho útočníka z M_A do M_B (ten, čo má najväčší rozdiel kvalít $f_b - f_a$) a potom najlepšieho nehrajúceho útočníka (t.j. z M_D) presunieme do útoku (do M_A).

3. fáza: Počítajme optimálne $(A, B, i + 1)$ – družstvo, ak už poznáme optimálne (A, B, i) – družstvo $(0 \leq i < C)$. Podobne ako 2. fáza, len tých variánt bude 5 a budú o trochu komplikovanejšie.

Prvá je, že presunieme najlepšieho nehrajúceho obrancu do obrany.

Druhá je, že presunieme najvýhodnejšieho útočníka do obrany a potom najlepšieho nehrajúceho útočníka do útoku.

The fishes answered, with a grin,
„Why, what a temper you are in!“

Tretia je podobná druhej: Presunieme najvýhodnejšie poliara do obrany a potom najlepšieho nehrajúceho poliara do poľa.

Štvrtá je trojpresunová: Presunieme najvýhodnejšiehoho z poľa do obrany a potom najvýhodnejšieho z útoku do poľa a nakoniec najlepšieho nehrajúceho útočníka dáme do útoku.

Piata je podobná štvrtej: Presunieme najvýhodnejšieho z útoku do obrany a potom najvýhodnejšieho z poľa do útoku a nakoniec najlepšieho nehrajúceho poliara dáme do poľa.

Mnohí z vás prišli ešte na drobné vylepšenia, ktoré ale asymptotickú časovú zložitosť nevylepšia. Napríklad, že si udržujeme zotriedené zoznamy podľa jednotlivých kvalít, alebo že netreba prechádzať vždy všetkých hráčov, ale len príslušnú množinu.

I told them once, I told them twice:
They would not listen to advice.

Vyskúšanie ľubovoľnej varianty hravo spravíme v čase $O(N)$. Stačí hľadať niekoľko (najviac 3) najvýhodnejších presunov v najviac 5 variantách. Z nich sa v konštatnom čase vyberie najvýhodnejšia. Toto sa robí $A + B + C \leq N = O(N)$ krát. Celková časová zložitosť bude teda $O(N^2)$.

Pamätať si budeme vstup a priebežne si pamätáme aktuálne vyzerajúce družstvo, t.j. kto a kde hrá. Celková pamäťová zložitosť bude teda len $O(N)$.

Listing programu:

```
Program Futbal;

const MaxN  = 1000;
      Vela  = 12345; { velke cislo }

var N,A,B,C:integer;
    i,sum,s:integer;
```

[3] Prechdzajúci odstavec je klúčový na pochopenie riešenia. Odporúčam ho aspoň 3-krát prečítať a porozmýšľať.

```
    { h[i,1] - kvalita hraca v utoku, h[i,2] - v poli, h[i,3] - v obrane }
    { h[i,0]=0 (to je kvoli elegancii) }
    { I took a kettle large and new, }
    { Fit for the deed I had to do. }

    h:array[1..MaxN,0..3] of integer;

    { f[i] je post hraca i (0-nic, 1-utok, 2-pole, 3-obrana) }
    f,g:array[0..MaxN] of integer;

{ Hlada najlepsi presun zo skupiny x do skupiny y; x,y=0,1,2,3 }
Function Naj(x,y:integer; var max_hrac:integer):integer;
var j,max:integer;
begin
     if (x=y) then { specialny, nezaujimavy pripad }
     begin
          Naj:=0;
          max_hrac:=0;
          exit;
     end;

     max:=-Vela; max_hrac:=0;

     for j:=1 to N do
     if (f[j]=x) and (h[j][y]-h[j][x]>max) then
     begin
          max:=h[j][y]-h[j][x];
          max_hrac:=j;
     end;

     Naj:=max;
end;

Procedure Varianta(x,y,z:integer);
var ss,max1,max2,max3 :integer;
begin
     { vyhodnost varianty }
     ss:=Naj(0,x,max1) + Naj(x,y,max2) + Naj(y,z,max3);
     { My heart went hop, my heart went thump: }
     { I filled the kettle at the pump. }

     { ak je to lepsia varianta... }
     if ss>s then
     begin
          g:=f;        { ...tak si ju zapamatame }
          g[max1]:=x;
          g[max2]:=y;
          g[max3]:=z;
          s:=ss;
     end;
```

```
end;

begin
     assign(input,'futbal.in'); reset(input);
     assign(output,'futbal.out'); rewrite(output);

     { nacitame vstup }
     readln(A,B,C,N);
     for i:=1 to N do
     begin
          readln(h[i,1],h[i,2],h[i,3]);
          h[i,0]:=0;
     end;

     { 0. faza - optimalne riesenei pre druzstvo (0,0,0)}
     { t.j. nikto nikde este nehra }
     for i:=1 to N do f[i]:=0;
     sum:=0; { kvalita druzstva }

     { 1. faza - optimalne riesenie pre druzstvo (A,0,0) }
     { t.j. vyberame najlepsich A utocnikov }
     for i:=1 to A do
     begin
          s:=-Vela;
          Varianta(1,1,1);  { 1. (a jedina) varianta presunu hracov }
          f:=g; sum:=sum+s; { optimalne riesenie pre druzstvo (i,0,0) }
     end;

     { 2. faza - optimalne riesenie pre druzstvo (A,B,0) }
     for i:=1 to B do
     begin
          s:=-Vela;
          Varianta(2,2,2);  { 1. varianta }
          { Then some one came to me and said }
          { "The little fishes are in bed." }
          Varianta(1,2,2);  { 2. varianta }
          f:=g; sum:=sum+s; { optimalne riesenie pre druzstvo (A,i,0) }
     end;

     { 3. faza - optimalne riesenie pre druzstvo (A,B,C) }

     for i:=1 to C do
     begin
          s:=-Vela;
          Varianta(3,3,3);  { 1. varianta }
          Varianta(1,3,3);  { 2. varianta }
          Varianta(2,3,3);  { 3. varianta }
          Varianta(1,2,3);  { 4. varianta }
          Varianta(2,1,3);  { 5. varianta }
          f:=g; sum:=sum+s; { optimalne riesenie pre druzstvo (A,B,i) }
     end;
```

```
    writeln('Najlepsie druzstvo je: ',sum);
end.
```

5. O Santolande III

opravoval Rišo
(max. 15 bodov)

Ako to už býva v sérii príkladov o Santolande zvykom, príklad mal dve nezávisle hodnotené podúlohy. Ľahšiu podúlohu a) a ťažšiu podúlohu b). A aj keď ani jedna z nich nebola taká ťažká, ako sa na prvý pohľad mohlo zdať, len veľmi málo z vás sa ich (najmä časť b) pokúsilo vyriešiť.

I said to him, I said it plain,
„Then you must wake them up again.“

Za čo ste mohli získať body? Za časť a) ste mohli získať najviac 5 bodov. 5 bodov však mohla získať iba jaskyňa, ktorá obsahovala iba 4 komory. Za každú komoru navyše ste stratili 1 bod. Ak vami navrhnutá jaskyňa bola chybná, ale vaša základná myšlienka bola správna, stratili ste 2 body. 1 bod ste mohli stratiť za chýbajúci popis, resp. zdôvodnenie funkčnosti jaskyne.

Za časť b) ste mohli získať najviac 10 bodov. 1 bod ste mohli stratiť za chybný odhad časovej a pamäťovej zložitosti vášho programu a ďalší bod ste mohli stratiť za priveľkú pamäťovú zložitosť riešenia (ak bola pamäťová zložitosť väčšia ako dĺžka vstupu).

Vzorové riešenie časti a): Zopakujme si základné vedomosti o deliteľnosti: Číslo je deliteľné šiestimi práve vtedy , ak je deliteľné troma a dvoma zároveň. Troma je číslo deliteľné práve vtedy, ak je jeho ciferný súčet deliteľný troma. Dvoma je číslo deliteľné práve vtedy, ak je jeho posledná cifra párna.

I said it very loud and clear:
I went and shouted in his ear.

Naša jaskyňa bude mať 4 komory. Zostrojíme ju tak, že z každej komory bude viesť na jednu cifru práve jeden tunel.[4] Ak sa bude zlatokop nachádzať v niektorej z komôr, jeho poloha bude hovoriť niečo o deliteľnosti tej časti čísla, ktorú už prečítal:

- Ak sa zlatokop nachádza v komore č.1, časť čísla, ktorú prečítal, je deliteľná šiestimi
- Ak sa zlatokop nachádza v komore č.2, časť čísla, ktorú prečítal, je deliteľná tromi ale nie dvomi
- Ak sa zlatokop nachádza v komore č.3, časť čísla, ktorú prečítal, dáva po delení 6 zvyšok 1
- Ak sa zlatokop nachádza v komore č.4, časť čísla, ktorú prečítal, dáva po delení 6 zvyšok 2

Tunely zostrojíme tak, aby prechod nimi tieto vlastnosti zachovával. Ľahko sa dá zistiť, že ich musíme zostrojiť takto:

$(1,1,0)$	$(1,3,1)$	$(1,4,2)$	$(1,2,3)$	$(1,3,4)$	$(1,4,5)$	$(1,1,6)$	$(1,3,7)$
$(1,4,8)$	$(1,2,9)$	$(2,1,0)$	$(2,3,1)$	$(2,4,2)$	$(2,2,3)$	$(2,3,4)$	$(2,4,5)$
$(2,1,6)$	$(2,3,7)$	$(2,4,8)$	$(2,2,9)$	$(3,3,0)$	$(3,4,1)$	$(3,1,2)$	$(3,3,3)$
$(3,4,4)$	$(3,2,5)$	$(3,3,6)$	$(3,4,7)$	$(3,1,8)$	$(3,3,9)$	$(4,4,0)$	$(4,2,1)$
$(4,3,2)$	$(4,4,3)$	$(4,1,4)$	$(4,3,5)$	$(4,4,6)$	$(4,2,7)$	$(4,3,8)$	$(4,4,9)$

But he was very stiff and proud:
He said, „You needn't shout so loud!“

[4] Teda jaskyňa bude zodpovedať deterministickému konečnému automatu.

Začínať môžme v komore č.1. Ak totiž zlatokop neprečítal nijakú časť čísla, je to rovnaká situácia, ako keby prečítal 0. Rum treba zjavne umiestniť len do komory č.1.

Jaskyňa je skonštruovaná tak, že s daným papierikom má zlatokop len jednu možnosť, ako sa pohybovať. Tunely sú skonštruované tak, že po prečítaní určitej časti čísla na papieriku sa zlatokop nachádza v komôrke číslo 1 práve vtedy, ak je prečítaná časť čísla deliteľná 6. Po prečítaní celého čísla sa teda zlatokop nachádza v komôrke č.1 (čo je jediná komôrka s rumom) práve vtedy, ak bolo celé číslo deliteľné 6. Jaskyňa je teda korektná, vyhovuje zadaniu.

Vzorové riešenie časti b): Predpokladajme, že máme dané jaskyne A a B. Jaskyňu C zostrojíme tak, že jej komory budú prislúchať dvojiciam komôr (a, b), kde a je komora jaskyne A a b je komora jaskyne B. Tunely zostrojíme tak, aby platila nasledovná vlastnosť: Zlatokop sa môže prečítaním nejakej časti papierika (označme túto časť w) dostať v jaskyni C do komory (a, b) práve vtedy, ak w je zloženinou nejakých slov u a v takých, že v jaskyni A sa môže dostať prečítaním u do komory a a v jaskyni B sa môže dostať prečítaním v do komory b.

Pohyb zlatokopa po jaskyni C bude teda prebiehať tak, ako keby sa zlatokop hýbal súčasne v jaskyni A aj B. Pri prečítaní určitého písmena si pri tom môže vybrať, v ktorej jaskyni sa má pohnúť. V druhej jaskyni zostane stáť.

Zlatokop musí v novej jaskyni začínať v komore $(1, 1)$ (po neprečítaní ničoho sa v jaskyni A aj v jaskyni B môže zlatokop nachádzať iba v komore 1). Rum umiestnime do všetkých komôr (a, b) takých, že a je rumová komora jaskyne A a b je rumová komora jaskyne B. Potom z našej podmienky vyplýva, že po prečítaní celého papierika sa zlatokop nachádza v rumovej komore práve vtedy, ak slovo na papieriku vzniklo zmiešaním rumovej postupnosti jaskyne A a jaskyne B. Jaskyňa teda bude vyhovovať zadaniu.

A ako zostrojiť tunely? Nech v jaskyni A vedie tunel z a do a' na písmeno z a v jaskyni B vedie tunel z b do b' na písmeno z. Predstavme si situáciu, že zlatokop sa nachádza v komore (a, b) a má prečítať písmeno z z papierika. Toto písmeno mohlo patriť buď postupnosti pre jaskyňu A alebo postupnosti pre jaskyňu B. Teda tunel môže viesť buď do (a', b) alebo do (a, b'). Nijaký iný tunel na písmeno z v jaskyni C z (a, b) viesť nemôže.

And he was very proud and stiff:
He said „I'd go and wake them, if –“

Keď zostrojíme tunely podľa tohoto pravidla, prechod tunelmi bude zachovávať nami zvolenú vlastnosť. Jaskyňa bude teda vyhovovať zadaniu.

Implementácia je jednoduchá. Komory jaskyne C budeme číslovať nasledovne: Nech n_1 je počet komôr jaskyne A a n_2 je počet komôr jaskyne B. Potom komore (a, b) pridelíme číslo $(a-1)n_2 + b$. Takýmto číslovaním očíslujeme komory jaskyne C od 1 do $n_1 n_2$. Pre každý tunel jaskyne A vytvoríme n_2 tunelov jaskyne C podľa nášho pravidla. Analogicky, pre každý tunel jaskyne B vytvoríme n_1 tunelov jaskyne C. Vypisovať vytvárané tunely môžme priamo pri načítavaní pôvodných tunelov. Rumové komory jaskyne A budeme musieť načítať do pomocného poľa. Potom pri načítavaní rumových komôr jaskyne B budeme priamo vypisovať rumové komory jaskyne C.

Pamäťová zložitosť nášho riešenia je teda $O(n_1)$ – musíme si pamätať rumové komory jaskyne A. Časová zložitosť výpisu rumových komôr je $O(n_1 n_2)$ a časová zložitosť výpisu tunelov je $O(n_1 m_2 + n_2 m_1)$, kde m_1 je počet tunelov prvej a n_2 je počet tunelov druhej jaskyne. Celková časová zložitosť algoritmu teda je $O(n_1 n_2 + n_1 m_2 + n_2 m_1)$.

I took a corkscrew from the shelf:
I went to wake them up myself.

Listing programu:

```
program O_Santolande_III;

var n1,n2,m1,m2,r1,r2:integer;    { Udaje o vstupnych jaskyniach }
    k,l,i,j:integer;
    A:array[1..100] of integer;   { Rumove komory 1. jaskyne }
    c:char;

{Prevod dvojice komor v starych jaskyniach na cislo komory v novej jaskyni}
function Cislo(i,j:integer):integer;
begin
 Cislo:=(i-1)*n2+j;
 { And when I found the door was locked, }
 { I pulled and pushed and kicked and knocked. }
end;

begin
 readln(n1,m1,r1);   { pocet komor, tunelov a komor s rumom 1. jaskyne }
 readln(n2,m2,r2);   { pocet komor, tunelov a komor s rumom 2. jaskyne }
 writeln(n1*n2,' ',n1*m2+n2*m1,' ',r1*r2); { Vypis pocet komor, tunelov }
                     { a komor s rumom vyslednej jaskyne }
 for i:=1 to m1 do
  begin
   readln(c,k,l);      { Nacitaj tunel v 1. jaskyni }
   for j:=1 to n2 do   { Vypis vzniknute tunely v novej jaskyni }
     writeln(c,' ',Cislo(k,j),' ',Cislo(l,j));
  end;

 for i:=1 to r1 do readln(A[i]);  { Nacitaj rumove komory 1. jaskyne }

 for i:=1 to m2 do
  begin
   readln(c,k,l);      { Nacitaj tunel v 2. jaskyni }
   for j:=1 to n1 do   { Vypis vzniknute tunely v novej jaskyni }
     writeln(c,' ',Cislo(j,k),' ',Cislo(j,l));
  end;

 for i:=1 to r2 do
  begin
   readln(k);          { Nacitaj rumovu komoru 2. jaskyne }
   for j:=1 to r1 do write(Cislo(A[j],k),' '); { vzniknute rumove komory }
  end;
 writeln;
end.
```

And when I found the door was shut,
I tried to turn the handle, but–

Výsledková listina po 3. sérii kategórie KSP

	Meno a priezvisko	Škola	31	32	33	34	35	Σ
1.	Bella Peter	Gym. Jura Hronca Bratislava	14	10	14	15	14	67
2.	Záthurecký Tomáš	Gym. V. P. Tótha Martin	9	11	14	15	15	64
3.	Pokorný Michal	Gym. Grösslingová Bratislava	15	11	14	7	12	59
4.	Tesař Marek	Gym. B. S. Timravy Lučenec	11	11	14	15	6	57
5.	Gaži Peter	Gym. Grösslingová Bratislava	10	11	14	4	15	54
6.	Tvarožek Jozef	Gym. Jura Hronca Bratislava	15	10	15	10		50
7.	Juhos Pavol	Gym. Grösslingová Bratislava	9	11	15	1	13	49
8.	Katrenič Ján	Gym. Spišská Nová Ves	10	11	12	12	3	48
9.	Veselý Jozef	Gym. Nitra, Golianova	8	10	14	10	4	46
10.	Záhorec Tomáš	Gym. Jura Hronca Bratislava	2	10	15	14		41
	Bauer Radovan	Gym. Košice, Poštová	11	10	12	5	3	41
12.	Glaus Michal	Gym. Jura Hronca Bratislava	10	11	14		5	40
13.	Mikuš Michal	Gym. Jura Hronca Bratislava		10	15	14		39
14.	Dzetkulič Tomáš	Gym. P. Horova Michalovce	10	9	14	0	5	38
15.	Glaus Peter	Gym. Jura Hronca Bratislava	9	11	13		4	37
16.	Steskal Ľuboš	Gym. Grösslingová Bratislava	10	10	8		7	35
	Dzurjanin Peter	Gym. Grösslingová Bratislava	10	10	15			35
18.	Čulák Radovan	Gym. Nitra, Golianova	8	7	11	7		33
19.	Miklian Peter	Gym. B. S. Timravy Lučenec	2	11	15	1		29
20.	Mazák Ján	Gym. Košice, Poštová	5	8	5	4	5	27
21.	Ludha Marek	Gym. Tajovského Banská Bystrica		11	13			24
22.	Sághy Tomáš	SPŠE Nové Zámky	11	10				21
	Mrázik Martin	Gym. Jura Hronca Bratislava		8	13			21
24.	Kucharík Ladislav	SPŠE Piešťany	2	6	12			20
25.	Konečný Štefan	Gym. Jura Hronca Bratislava	1	7	9			17
26.	Paulíny Kamil	Gym. Košice, Alejová	9	7				16
27.	Berta Radovan	Gym. Vranov nad Topľou	1	7			3	11
	Dvorský Marián	Gym. Košice, Šrobárova		11				11
	Plavčan Juraj	Gym. Šk. bratov Bratislava	1	10				11
30.	Danko Ondrej	Gym. Jura Hronca Bratislava	nn	10	nn			10
31.	Bednárik Ľuboš	Gym. Ľudovíta Štúra Trenčín		9				9
32.	Rybár Pavol	Gym. Šk. bratov Bratislava		7				7
33.	Rudišin Miroslav	Gym. Košice, Šrobárova					5	5
34.	Rjaško Michal	Gym. Vranov nad Topľou	nn	3				3
35.	Ochránek Martin	Gym. Jura Hronca Bratislava	0			1		1