Korepondenčný seminár z programovania
XIX. ročník, 2001/2002
Katedra vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

KSP finančne podporuje nadácia Open Society Fund Bratislava a
IUVENTA – zariadenie pre vožný čas detí, mláde«e a dospelých
MICROSTEP spol s.r.o.

Vzorové riešenia 2. kola zimnej časti

Milé deti,

Vianoce máme úspešne za sebou, a to okrem iného znamená, že nadišiel ten správny čas na vzorové riešenia druhého kola. Nemusíte sa ich vôbec báť – podľa našich doterajších skúseností nehryzú. Tak na čo ešte čakáte? Tieto riešenia sú tu len a len pre vás...

Chemist who falls in acid is absorbed in work.

KSPáci

1. Ochrana zdravia pri práci

opravovalo YoYo δ
(max. 15 bodov)

Po tom, čo Kontrolná Skupina Pracovníkov dostala vaše programy, rozhodla sa, že by bolo najlepšie delegovať jedného z nich, aby sa tou kopou prehrabal a vybral ten najrýchlejší. No a keďže delegovaný pracovník stále nemá funkčné stopky[1], rozhodol sa roztriediť ich podľa nasledovných kritérií. No a výsledok bol takýto:

- $O(N \log N)$ – či už priemerne alebo v najhoršom prípade, išlo väčšinou o ten najjednoduchší prístup, teda utriediť a vybrať k-ty najmenší prvok. Bodové hodnotenie: **9 a menej bodov**.
- $O(N)$ v priemernom prípade – quicksortové hľadanie mediánu – **13 bodov**.
- $O(N)$ v najhoršom prípade – median-of-five – **15 bodov**.
- $O(N)$ s predpokladom, že veky sú celé čísla z nejakého intervalu, najčastejšie $0 \ldots 120$ alebo $0 \ldots 200$. Keďže žiadne z týchto vecí neboli zadaním zaručené, dalo sa za ne získať maximálne **7 bodov**.

Pozrime sa teraz na vaše riešenia. V každom sa vyskytovala funkcia hľadajúca k-ty najmenší prvok a nevyzerala na pohľad pekne. Predpokladajme zatiaľ, že túto funkciu už máme dispozícii a nech sa volá napríklad `Select`. Potom ale vôbec nie je ťažké vyriešiť náš príklad. Najprv zavoláme `Select` pre $k = (n+1)$ `div` 2. Tým získame medián, označme si ho m. Veky pracovníkov teraz zmeníme na ich odchýlky od m, teda $v_i := |v_i - m|$. Teraz znovu zavoláme `Select` pre správne k a tým získame k-teho pracovníka v poradí s postupne sa zväčšujúcim rozdielom vekov od „mediána“. Už nám stačí iba raz prejsť celé pole a vypísať tých k „najstrednejších“ pracovníkov. Celý tento postup by bol lineárny od počtu pracovníkov, ak by sme vedeli v lineárnom čase zrealizovať aj `Select`.

Ak nám stačí, aby to bolo v priemere lineárne, nie je to až také ťažké. Môžeme použiť podobný algoritmus ako quicksort. Zvolíme si nejaký prvok, pole preusporiadame tak, aby naľavo boli prvky od neho menšie, potom rovné a následne väčšie. A tu potom nastáva jediný rozdiel oproti quicksortu. Ten sa totiž rekurzívne zavolá na prvú a tretiu časť. My však teraz hľadáme len k-ty prvok. Pre ten ale vieme povedať, do ktorej z týchto troch častí padol. Ak do druhej, hotovo. Ak do prvej alebo tretej, rekurzívne sa zavoláme **len** na ňu.

[1] a aj keby ich mal, nemá trpezlivosť čokoľvek s nimi merať

Takto v priemernom prípade prvýkrát preusporiadame N prvkov, pri druhom $\frac{N}{2}$, atď. Takýto algoritmus má teda v priemernom prípade časovú zložitosť $O(N + \frac{N}{2} + \frac{N}{4} \ldots) = O(N)$.

Ako by sa to ale dalo urobiť tak, aby to bolo lineárne aj v najhoršom prípade? Ide o modifikáciu uvedeného postupu s názvom median-of-five, ktorý má aj v najhoršom prípade časovú zložitosť $O(N)$. Z pedagogicko-metodologických príčin však tento algoritmus neuvádzame, záujemcovia si ho môžu pohľadať napr. v knihe *Cormen-Leiserson-Rivest: Introduction to Algorithms*.

Listing programu:

```
program Ochrana_zdravia_pri_praci;
const maxN=100;
var a:array[1..maxN] of integer;
    v:array[1..maxN] of real;
    n,k,i,j:integer;
    median,kty:real;

function Select(k,l,r:integer):integer;
var p:real;
    i,j,t:integer;
begin
    p:=v[a[(l+r) div 2]];
    i:=l; j:=r;
    repeat
      while v[a[i]]<p do inc(i);
      while v[a[j]]>p do dec(j);
      if (i<=j) then begin
        t:=a[i];  a[i]:=a[j];  a[j]:=t;
        inc(i);
        dec(j);
      end;
    until i>j;

    if (k>j) and (k<i) then
      Select:=a[k]
    else
    if (k<=j) then
      Select:=Select(k,l,j)
    else
      Select:=Select(k,i,r);
end;


begin
  write('Pocet zamestnancov: ');
  readln(n);
  write('k : ');
  readln(k);
  for i:=1 to n do begin
    write('Vek ',i,'. zamestnanca: ');
    readln(v[i]);
    a[i]:=i;
```

```
  end;
  median:=v[Select( (n+1) div 2,1,n)];
  for i:=1 to n do v[i]:=abs(v[i]-median);
  kty:=v[Select( k, 1, n)];
  Write('Pracovnici: ');
  i:=0; j:=0;
  while( (i<n) and (j<k)) do begin
    inc(i);
    if v[a[i]]<=kty then begin write(a[i],'  '); inc(j); end;
  end;
  writeln;
end.
```

2. O ťavích dostihoch

opravoval WSX
(max. 18 bodov)

Vedenie stávkovej spoločnosti bolo s vami spokojné. Vaše riešenia sa dajú rozdeliť podľa časovej a pamäťovej zložitosti do niekoľkých skupín:

- Čas $O(NK)$, pamäť $O(1)$: **18 bodov**.
- Čas $O(NK)$, pamäť $O(K)$: Založené na dynamickom programovaní. Bodová hranica **15 bodov**.
- Čas $O(NK)$, pamäť $O(NK)$: Používajú rekurentný vzťah, **13 bodov**.
- Polynomiálny čas horší ako $O(NK)$: Dalo sa získať **10 bodov**.
- Exponenciálne riešenia: Podľa optimalizácie, maximálne **7 bodov**.
- Nefungujúce riešenia: Maximálne **2 body**.

Za nedostatočný popis (hlavne za chýbajúci dôkaz správnosti) sa dali stratiť **3 body**.

18 bodové riešenie sa nevyskytlo. Takéto riešenie je založené na vzťahu, ktorý odvodili Knuth a Netto. Vzhľadom na komplikovanosť použitej matematickej teórie ho prenecháme[2] matematikom.

Ukážeme si, ako spraviť riešenie používajúce pamäť $O(K)$. Na to budeme potrebovať nejakú matematickú teóriu, ktorú si odvodíme. Nech dôjdu ťavy do cieľa v poradí p_1, p_2, ... p_N. Pod pojmom počet inverzií permutácie p budeme rozumieť počet takých dvojíc i, j v permutácii, že $i < j$ a zároveň $p_i > p_j$. Predbiehajúca ťava má vždy vačšie číslo, v opačnom prípade by sa už museli niekedy navzájom predbehnúť, čo je spor so zadaním. Preto pri každom predbiehaní sa počet inverzií zvýši o 1. Keďže pri štare je počet inverzií 0, počet inverzií je zhodný s počtom predbehnutí. Nech $A(N, K)$ je počet permutácií veľkosti N s K inverziami. Takáto permutácia vznikne, keď do permutácie dĺžky $N-1$ pridáme prvok s hodnotou N. Pridaním tohto prvku na pozíciu I sa počet inverzií zvýši o $N-I$. Potom pre hodnotu $A(N, K)$ dostaneme rekurentný vzťah: $A(N,K) = \sum_{I=1}^{N} A(N-1, K-(N-I)) = \sum_{I=0}^{N-1} A(N-1, K-I) = A(N-1,K) + \sum_{I=1}^{N-1} A(N-1,K-I)$. Potom pre $A(N, K-1)$ platí vzťah: $A(N,K-1) = \sum_{I=0}^{N-1} A(N-1,K-I-1) = A(N-1,K-N) + \sum_{I=1}^{N-1} A(N-1,K-I)$. Porovnaním vzťahov dostaneme rekurentný vzťah: $A(N,K) = A(N-1,K) + A(N,K-1) - A(N-1,K-N)$. Ďalej vieme, že $(\forall I)A(I,0) = 1$ (I-prvková permutácia bez inverzií je len jedna). Vzorové riešenie postupne vypĺňa tabuľku o rozmerov $N \times K$ a používa myšlienku dynamického programovania – novú hodnotu vypočítame pomocou už vypočítaných hodnôt. Tabuľku budeme vypĺňať zaradom pre I od 1 po N. Keďže podľa rekurentného vzťahu potrebujeme vždy len hodnoty v riadkoch N a $N-1$, stačí si pamätať len posledné 2 riadky.

[2] odvodenie možno nájsť na http://academic.csuohio.edu/bmargolius/homepage/inversions/invers.htm

Pri vypĺňaní riadku najskôr nastavíme hodnotu $A(I, 0)$, ostatné hodnoty vypĺňame podľa vzťahu pre J od 1 po K. Tu si treba uvedomiť, že v prípade $J < I$ by sme sa podľa vzťahu odvolávali na počet permutácií so záporným počtom inverzií. Keďže takýchto permutácií je 0, môžme v tomto prípade vynechať daný člen. Výsledkom je hodnota $A(N, K)$, ktorá udáva počet permutácií (spôsobov predbiehania) veľkosti N prvkov (tiav) s K inverziami (predbehnutiami).

Odhad zložitosti: Na vyplnenie jedného políčka tabuľky potrebujeme čas $O(1)$. Keďže vypĺňame tabuľku rozmerov $N \times K$, potrebujeme čas $NK.O(1) = O(NK)$. Každý riadok tabuľky zaberá $O(K)$ miesta, potrebujeme si pamätať 2 riadky, čiže pamäťová zložitosť je $O(K)$.

Listing programu:

```
Const N=3;
      K=2;

Var A:Array[0..1, 0..K] Of LongInt;
    I, J:LongInt;

Begin
  A[1, 0]:=1; { jednoprvkova mnozina bez inverzii }
  For I:=1 To K Do A[1, I]:=0; { s inverziami (nemozne) }
  For I:=2 To N Do
  Begin
    A[I And 1, 0]:=1; { bez inverzii }
    For J:=1 To K Do
    Begin
      A[I And 1, J]:=A[I And 1, J-1]+A[(I-1) And 1, J];
      If J>=I Then Dec(A[I And 1, J], A[(I-1) And 1, J-I]);
    End;
  End;
  WriteLn(A[N And 1, K]);
End.
```

3. O Martowovi II

opravoval JanoS
(max. 15 bodov)

Tento príklad sa dal riešiť dvoma spôsobmi. Ten prvý napadol každého z vás a spočíval v odmeraní vzdialenosti medzi každými dvoma zastávkami. Samozrejme takéto riešenie malo časovú zložitosť $O(D.N^2)$, čo nie je príliš pozitívne, a bolo ohodnotené max. **8 bodmi**. Druhý spôsob, nazvime ho vzorový, pracuje v časovej zložitosti $O(N.2^D)$ (uvedomte si, že toto je výrazne lepšie, nakoľko predpokladáme že $D \ll N$) a bol ohodnotený maximálnym počtom **bodov 15**. Nejaký ten bodík – dva som strhol za nekompletný popis, žiadny popis alebo zlé odhady zložitosti.

Najprv jednu definíciu na zahriatie[3]:

Def.: Manhattanovská vzdialenosť dvoch bodov v D-rozmernom priestore je $\sum_{i=1}^{D} |y_i - x_i|$.

Áno, správne tušíte, vzdialenosť dvoch autobusových zastávok v našej úlohe je presne Manhattanovská vzdialenosť. Aby sme sa dostali k myšlienke vzorového riešenia, zjednodušme si o čosi úlohu. Predpokladajme, že $D = 1$ a teda sú všetky body na jednej priamke. Ako teraz nájdeme najvzdialenejšie? Nájdeme najľavejší a najpravejší a máme ich. Prenesme

[3] bola tuhá zima

sa do dvojrozmerného priestoru, čiže do roviny. Keď sa pokúsime opísať týmto bodom obdĺžnik, ktorého strany zvierajú s osami uhol 45° (podobne ako sa opisuje kružnica, teda nájdeme najmenší obdĺžnik, do ktorého patria všetky body), zistíme, že tie dva najvzdialenejšie ležia na jeho náprotivným stranách. Ak budeme postupovať takto ďalej, zistíme že v D-rozmernom priestore ležia dva manhattanovsky najvzdialenejšie body na povrchu D-rozmerného hyperkvádra, ktorý má steny pootočené o 45 stupňov voči osiam.

Skúsme to zapísať formálnejšie. Vzdialenosť, ktorú chceme maximalizovať, je $|x_1 - y_1| + |x_2 - y_2| + \cdots + |x_D - y_D|$. Predpokladajme teraz, že platí $(\forall i) x_i \geq y_i$. Našu vzdialenosť si môžeme zapísať ako $(x_1 - y_1) + (x_2 - y_2) + \cdots + (x_D - y_D)$ alebo ako $(x_1 + x_2 + \cdots + x_D) - (y_1 + y_2 + \cdots + y_D)$. Body s maximálnou takouto hodnotou však vieme poľahky nájsť – stačí nájsť bod s maximálnym súčtom súradníc a bod s minimálnym súčtom, a tie vieme nájsť v čase lineárnom od počtu bodov. Problémom teraz je náš predpoklad. V prípade že by pre nejaké i platilo $x_i < y_i$, museli by sme v našom výpočte obrátiť znamienko i-teho člena. Koľko kombinácií znamienok existuje? Pre každé máme dve možnosti, plus a mínus, a preto to musíme otestovať 2^D-krát, pre všetky kombinácie D znamienok.

Na dôkaz správnosti nášho algoritmu už len potrebujeme dokázať, že tá správna dvojica sa vyskytne aspoň raz ako minimálny a maximálny súčet počas 2^D pokusov. Uvažujme taký pokus, kde priradenia znamienok vyhovujú tejto dvojici. Keďže táto dvojica reprezentuje najvzdialenejšie body, žiaden iný bod nemôže mať tento výraz menší ako menší z dvojice, alebo väčší ako ten väčší z dvojice. Povedzme, že by sme nejakým bodom zvýšili maximum. Tento bod by tým pádom bol vzdialenejší od minima viac ako ten pôvodný, čo je spor s predpokladom, že sú najvzdialenejšie. Takže z toho vyplýva, že na optimálne riešenie určite naďabíme.

Takéto riešenie má časovú zložitosť $O(N.2^D)$, čo je v porovnaní s triválnym riešením $O(N^2.D)$ lepšie v prípade, že $D < log_2 N$.

Listing programu:

```
Program O_Martowovi_II;

Var A:Array[1..100,0..10] Of Integer;
    I,J,K,N,D,Min,Max,Hodnota,
    Rozmer,PomRozmer,Dolna,Horna,Prva,Druha,Rekord:Integer;

Begin
  Readln(N,D);
  Rekord:=-1;
  For I:=1 To N Do
    For J:=1 To D Do
       Read(A[I,J]);      {Rozmer je kombinacia + a -     }
  For Rozmer:=0 To (1 Shl (D-1))-1 Do Begin {Toto je 2^D-1}
    Min:=Maxint;
    Max:=-Maxint;
    For I:=1 To N Do Begin
      Hodnota:=0;
      PomRozmer:=Rozmer;
      For J:=1 To D Do Begin {Skontrolujeme posledny bit}
        If Odd(PomRozmer) Then Inc(Hodnota,A[I,J])
                          Else Dec(Hodnota,A[I,J]);
        PomRozmer:=PomRozmer Shr 1; {Bitova rotacia doprava}
      End; {Ak  je hodnota rekordna, je dobre si ju zapisat}
      If Min>Hodnota Then Begin Min:=Hodnota; Dolna:=I; End;
```

```
      If Max<Hodnota Then Begin Max:=Hodnota; Horna:=I; End;
    End;
    If Rekord<Max-Min Then Begin {Porovname dva krajne body}
       Rekord:=Max-Min;
       Prva:=Dolna; Druha:=Horna;
    End;
  End;
  Writeln(Rekord);
End.
```

opravoval MišoF
(max. 20 bodov)

4. O podnikovej sieti

Ták, opäť sme mali jednu úlohu, za ktorú sme dali maximum viac ako **15 bodov**. Ale nezúfajte, nebolo to zas až také zlé. Posadajte si pekne ku mne okolo ohníčka a ja vám porozprávam, ako sa to malo riešiť. A zoberte si niečo na jedenie, aby ste nemuseli od čítania odbiehať, keď vyhladnete, bude to trošku dlhšie...

Základom úspechu bolo pochopiť zadanie a preložiť si ho do reči teórie grafov. Počítače budú tvoriť vrcholy grafu, káble medzi nimi hrany. Počet sieťových kariet v počítači zodpovedá stupňu vrcholu. Keď už máme túto predstavu, nie je ťažké preložiť si zadanie úlohy – pýtame sa, či existuje (jednoduchý neorientovaný) súvislý graf, ak máme dané stupne jeho vrcholov. O tom nám niečo hovorí Havlova veta. Presnejšie:

Veta 1 (Havel): Nech $n > s_1 \geq \ldots \geq s_n \geq 0$, potom ($n$-vrcholový) graf so stupňami vrcholov $s_1, \ldots, s_n$ existuje práve vtedy, keď existuje (($n-1$)-vrcholový) graf so stupňami vrcholov $s_2 - 1, \ldots, s_{s_1+1} - 1, s_{s_1+2}, \ldots, s_n$. (Inými slovami, nič nepokazíme, keď zoberieme vrchol najväčšieho stupňa a spojíme ho s toľkými ďalšími vrcholmi najväčšieho stupňa, aký má mať stupeň.)

My si túto vetu dokážeme v trochu silnejšom znení.

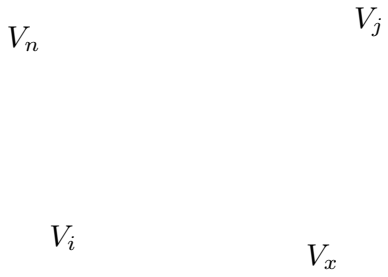
Veta 2 (MišoF.): Nech $n > s_1 \geq \ldots \geq s_{n-1} \geq 0$, potom ($n$-vrcholový) graf, ktorého vrcholy $v_1, \ldots, v_{n-1}$ majú stupne $s_1, \ldots, s_n$ existuje práve vtedy, keď existuje (($n-1$)-vrcholový) graf so stupňami vrcholov $s_1 - 1, \ldots, s_k - 1, s_{k+1}, \ldots, s_{n-1}$, kde $k = \deg(v_n)$. (Inými slovami, nič nepokazíme, keď zoberieme ľubovoľný vrchol a spojíme ho s toľkými inými vrcholmi najväčšieho stupňa, aký má mať stupeň.)

Dôkaz: Spätná implikácia (ak existuje menší, tak existuje väčší) je triviálna – pridáme vrchol a z neho hrany do správnych vrcholov. Opačnú sporom. Veta vlastne hovorí, že ak sa graf dá nejako zostrojiť, tak sa dá zostrojiť tak, že vrchol v_n je spojený s vrcholmi $v_1, \ldots, v_k$. Predpokladajme teda, že existuje graf, pre ktorý to neplatí. Zoberme[4] spomedzi všetkých možných spôsobov, ako ho zostrojiť, taký, kde je v_n spojený s najviac vrcholmi spomedzi $v_1, \ldots, v_k$.

Ak má v_n stupeň 0, tvrdenie zjavne platí. Preto nech $k = \deg(v_n) > 0$.

Nech i je najmenšie také, že v_n nie je spojené s v_i, zjavne $i \leq k$. Nech $j > k$ je ľubovoľné také, že v_n je spojené s v_j. Také i a j zjavne existujú a takisto zjavne má v_j nanajvýš taký stupeň ako v_i. Ale v_j je spojené s v_n a v_i nie je, preto má v_i viac susedov medzi ostatnými vrcholmi (všetkými okrem v_i, v_j, v_n) ako v_j. Potom ale určite existuje vrchol v_x taký, že v_i je spojené s v_x, ale v_j nie je.

[4] Toto je klasická finta, ako korektne zapísať dôkaz štýlu „keď spravíme toto, tak sa to zlepší a po konečnom počte opakovaní dostaneme žiadaný výsledok.“ Takisto aj tu by sa dalo postupne prepájať vrcholy – rozmyslite si, ako.



Našli sme teda dve dvojice vrcholov – v_i je spojené s v_x, ale nie s v_n, v_j je spojené s v_n, ale nie s v_x. Zostrojme nový graf, v ktorom namiesto hrán v_iv_x a v_jv_n budú hrany v_iv_n a v_jv_x. V tomto grafe budú mať všetky vrcholy rovnaké stupne ako v pôvodnom, ale vrchol v_n bude spojený s viac vrcholmi spomedzi $v_1, \ldots, v_k$ ako v pôvodnom grafe, čo je hľadaný spor.[5]

No a ďalej už je to jednoduchšie. Keď máme dané stupne vrcholov, vieme podľa predchádzajúcej vety zistiť, či existuje graf s danými stupňami vrcholov jednoduchým algoritmom – zakaždým zoradíme vrcholy podľa stupňov, zoberieme ľubovoľný z nich (napr. ten s najmenším alebo najväčším stupňom) a skúsime ho spojiť s toľkými ostatnými s najväčšími stupňami, aký má stupeň. Ak sa to nedá (nie je ich dosť), vieme, že taký graf neexistuje. Ak sa to dá, dostali sme graf s menej vrcholmi, ktorý sa dá zostrojiť práve vtedy, keď sa dá zostrojiť pôvodný graf. Preto opäť spustíme náš algoritmus na tento menší graf. Keďže pri každom kroku sa počet vrcholov zmenší, je tento postup konečný.

Pokiaľ teda žiadny graf s danými stupňami vrcholov neexistuje, zjavne neexistuje ani súvislý graf s týmito stupňami vrcholov. Čo ak nejaký graf existuje?

Určite žiaden vrchol nesmie mať stupeň 0. Takisto graf na to, aby bol súvislý, potrebuje mať aspoň $n-1$ hrán, takže súčet stupňov vrcholov musí byť aspoň $2n-2$. Ukážeme, že stačí, aby tieto podmienky boli splnené.

Veta 3: Ak existuje graf s danými stupňami vrcholov, všetky sú nenulové a ich súčet je aspoň $2n-2$, tak existuje aj súvislý graf s týmito stupňami vrcholov.

Dôkaz: Sporom. Majme postupnosť stupňov vrcholov, pre ktorú sú všetky podmienky splnené, ale každý graf s týmito stupňami vrcholov má aspoň 2 komponenty. Zoberme taký, ktorý má spomedzi všetkých z nich najmenej komponentov.[6] Keby mal každý komponent menej hrán ako vrcholov, mal by celý graf aspoň o 2 viac vrcholov ako hrán. Potom ale súčet stupňov by bol najviac $2n-4$, čo je spor. Preto niektorý komponent má aspoň toľko hrán ako vrcholov.

[5] Tu je spomínaný pes zakopaný. Keď máme ľubovoľný graf s danými stupňami, opakovaním tohoto postupu vieme dosiahnuť, že vrchol v_n bude spojený s čím ďalej, tým viac spomedzi vrcholov $v_1, \ldots, v_k$. A keď to už nepôjde, vieme, že je spojený práve s nimi, čo sme chceli dosiahnuť.

[6] Obľúbená finta...

Tento komponent potom určite obsahuje kružnicu. Odstráňme ľubovoľnú jej hranu uv, tým jeho súvislosť neporušíme. Ostali nám v ňom dva vrcholy, ktorým chýba jedna hrana. Zoberme ľubovoľný iný komponent a odstráňme z neho ľubovoľnú hranu xy. Tým dostaneme opäť dva vrcholy, ktorým jedna hrana chýba a možno sa nám tento komponent rozpadne na dva. Teraz pridajme hrany ux a vy. Tým dostaneme graf s takými istými stupňami vrcholov ako pôvodný a menej komponentmi (naše dva komponenty sme zjavne spojili do jedného), čo je spor.

Tým by sme mali funkčný algoritmus s časovou zložitosťou $O(n^2 \log n)$. Pri načítavaní overíme, či majú všetky vrcholy nenulový stupeň a či máme dosť hrán. Potom nám už stačí len overiť, či existuje nejaký graf s danými stupňami vrcholov. To urobíme vyššie uvedeným spôsobom – vždy zotriedime stupne vrcholov, jeden si vyberieme a odstránime ho. Toto opakujeme, kým nám ešte nejaké vrcholy ostanú, alebo kým nezistíme, že to nejde. Jedno opakovanie trvá $O(n \log n)$ triedenie, O(n) odstránenie vrcholu. Keďže pri každom opakovaní sa nám počet vrcholov zmenší, bude opakovaní $O(n)$, odtiaľ celková zložitosť $O(n.(n \log n)) = O(n^2 \log n)$.

Máme teda nejaké riešenie, poďme ho zlepšovať. Kritická časť predchádzajúceho algoritmu je triedenie. Ale stačí si uvedomiť, že triedime čísla od 1 do $n-1$, teda môžeme použiť Counting Sort so zložitosťou $O(n)$ – porátame si, koľko je vrcholov ktorého stupňa a vyplníme to do poľa. Iný možný prístup je priamo pre každý stupeň si pamätať, koľko vrcholov tohoto stupňa máme. Oba tieto prístupy vedú k riešeniu s časovou zložitosťou $O(n^2)$, za ktoré sa dalo získať 15 bodov.

Skôr, ako som vymyslel lineárne riešenie, mal som „brutálne“ riešenie v čase $O(n \log n)$, ktoré využívalo nasledovnú myšlienku: Zoberme si postupnosť $p_1, \ldots, p_h$ – počty vrcholov stupňa $1, \ldots, h$. Keď odstránime vrchol najväčšieho stupňa, zmení sa táto postupnosť len málo – najskôr p_h zmenšíme o 1, potom ju rozdelíme na dve časti. Vrcholom v „ľavej“ sa stupeň nezmení, tým v „pravej“ sa zníži o 1. To ale znamená, že novú postupnosť dostaneme tak, že sčítame „ľavú“ časť a o 1 doľava posunutú „pravú“ časť.

Príklad: Zoberme pôvodnú postupnosť 7,1,3,1,1,1,3 (t.j. 7 vrcholov stupňa 1, po 3 stupňov 3 a 7 a po 1 vrchole ostatných stupňov). Potom nová bude vyzerať nasledovne: (Odstránime vrchol stupňa 7, v pravej časti bude 7 vrcholov, v ľavej ostatné.)

pravá	`2 1 1 1 2`
ľavá	`7 1 1`
pravá posunutá	`2 1 1 1 2`
nová	`7 3 2 1 1 2`

Nie je ťažké všimnúť si, že nová postupnosť (min. o 1 kratšia) sa od pôvodnej líši len v poslednom člene a v okolí miesta, kde sa ľavá časť prekrýva s posunutou pravou. Keby sme si udržovali tieto hodnoty v poli, aj tak nám to nepomôže. Existuje ale prefíkaná dátová štruktúra, zvaná 2-3 strom, s ktorej pomocou sa dá jeden takýto krok spraviť v čase $O(\log n)$. Ale rozpisovať to tu radšej nebudem, aj tak je tento vzorák výnimočne dlhý.[7]

A keď som uvidel, že lineárne riešenie by malo príliš dlhý a zložitý dôkaz, povedal som si, že to bez neho nejako prežijete. Veď viete, ako sa hovorí, pridlhé vzorové riešenie by bolo z pedagogických dôvodov nevhodné.

A pre tých pár odvážlivcov, ktorí sa až sem dočítali, pár slov o tom, **ako som hodnotil vaše riešenia**. Poďme od najlepších.

- Riešenia v čase $O(n)$ mohli získať **20 bodov**.
- Riešenia v čase $O(n \log n)$ mohli získať **18 bodov**.
- Riešenia v čase $O(n^2)$ mohli získať **15 bodov**.
- Funkčné riešenia v horšom polynomiálnom čase mohli získať **12 bodov**.

[7] Ale keď sa nájdu záujemcovia, môžem o 2-3 stromoch porozprávať na sústredku. Jéj, mám nápad na prednášku :-)

- Funkčné riešenia v exponenciálnom čase mohli získať **8 bodov**, keby boli.
- Riešenia, ktoré len zistili, či graf existuje, mohli získať do **10 bodov**, výnimočne aj viac.
- Heuristiky[8] mohli získať do **5 bodov** podľa prefíkanosti.
- Iným spôsobom nefunkčné riešenia boli hodnotené individuálne podľa môjho skromného názoru, ako ďaleko mali k funkčnému a k akému dobrému.

Samozrejme sa dali stratiť body za chýbajúci, prípadne nedostatočný či nepochopiteľný popis, za chýbajúci dôkaz správnosti algoritmu a v neposlednom rade za chýbajúci program. A tiež za opisovanie!

Listing programu:

```
program ZistiCiJeTakySuvislyGraf;
{ Riesenie v O(n^2) }

const maxn = 1000;

var vrcholy,stupne : array[0..maxn] of integer;
    n : integer;

procedure Load;
var i,sum : integer;
begin
   readln(n); sum:=0;
   if (n=0) then begin writeln('DA SA.'); halt; end;
   for i:=1 to n do begin
      write(i,'. vrchol: '); readln(stupne[i]);
      inc(sum,stupne[i]);
      if (stupne[i]<=0) or (stupne[i]>=n) then begin
         writeln('NEDA SA.'); halt;
      end;
   end;
   if (sum mod 2=1) or (sum<(2*n-2)) then begin
      writeln('NEDA SA.'); halt;
   end;
end;

procedure CountSort;
var i,j : integer;
begin
   for i:=0 to n do vrcholy[i]:=0;
   for i:=1 to n do inc(vrcholy[stupne[i]]);
   j:=1;
   for i:=n downto 0 do begin
      while vrcholy[i]>0 do begin
         stupne[j]:=i;
         dec(vrcholy[i]);
         inc(j);
      end;
```

[8] program, ktorý pre väčšinu vstupov dá správnu odpoveď

```
   end;
end;

procedure Verify;
var i : integer;
begin
   while true do begin
      CountSort;
      if stupne[1]=0 then begin
         writeln('DA SA.'); halt;
      end;
      for i:=2 to stupne[1]+1 do begin
         if stupne[i]=0 then begin
            writeln('NEDA SA.'); halt;
         end;
         dec(stupne[i]);
      end;
      stupne[1]:=0;
   end;
end;

begin
   Load;
   Verify;
end.
```

5. O čiernych krabičkách II

opravoval Braňo
(max. 15 bodov)

Vaše riešenia sa dali rozdeliť na dve kôpky. Na jednej boli tie, ktorým operácia `ExtractMin` alebo `ExtractMax` trvala v priemere čas $O(n.\log n)$. Na druhej tie, ktorým každá operácia trvala v priemere $O(\log n)$. Riešenia som obodoval takto:

- Čas $O(\log n)$ a 4 krabičky – **13 bodov**
- Čas $O(n.\log n)$ a 2 krabičky – **7 bodov**
- Za pamäť $O(\log n)$ – $-$**1 bod**
- Za každú krabičku navyše – $-$**1 bod**
- Za snahy o optimalizáciu – max $+$**1 bod**
- Popis a dôkaz – $+$**2 body**

Vzorové riešenie: Na začiatku majme 2 čierne krabičky verzie 2 (ČK).

Prvá ČK bude iba obyčajná ČK, z ktorej vieme rýchlo vybrať minimálny prvok. Označme ju $MinKrab$.

Druhá ČK bude vedieť rýchlo vybrať najväčší prvok. Je to ČK, do ktorej budeme vkladať invertovné hodnoty (miesto x vložíme $-x$) a pri vyberaní znova invertujeme. Najmenšie invertované číslo medzi invertovanými je neinvertované najväčšie medzi neinvertovanými[9] ($x \geq y \iff -x \leq -y$). Označme ju $MaxKrab$.

Na začiatku sú všetky ČK prázdne. Keď chceme vložiť prvok x do našej pamäte, vložíme x do $MinKrab$ a $-x$ do $MaxKrab$. Nájsť najvačší alebo najmenší prvok potom vieme

[9] dobré zaklínadlo ...

jednoducho a rýchlo – iba vyberieme zo správnej krabičky. Problém je v tom, že ho vyberieme len z jednej ČK a v druhej ostane. Treba zaistiť, aby sme ho z druhej už nevybrali. Označme **živé** prvky tie, ktoré sú v oboch krabičkách a **mŕtve** tie, ktoré sú len v jednej. Keď vyberieme živý prvok, vypíšeme ho a on umrie. Keď vyberieme mŕtvy, iba ho zahodíme. Tento postup zaistí, že vypíšeme každý prvok práve raz.

Použijeme ďalšie dve pomocné ČK $MinMrtve$ a $MaxMrtve$. V nich budú tie hodnoty, ktoré sú už mŕtve. Algoritmus pre `ExtractMin` bude vyzerať takto: Vyberieme $a = ExtractMin(MinKrab)$ a $b = ExtractMin(MinMrtve)$. Ak $a = b$ (a je mŕtvy), tak ich zahodíme a opakujeme znovu. Ak sa nerovnajú, vypíšeme a užívateľovi, do $MinMrtve$ vložíme späť b (b je stále mŕtvy) a do $MaxMrtve$ pridáme $-a$ ($-a$ umrel). `ExtractMax` podobne[1]0. A test na prázdnosť pamäte? Keď sa zbavíme mŕtvych prvkov na vrchu $MinKrab$ a $MaxKrab$, tak sa stačí pozrieť, či sú obe prázdne.

Spočítajme, koľko práce nám dá spracovanie jedného prvku, keď ich je spolu n. Operácia $Insert$ trvá $2.O(\log n) = O(\log n)$, čo je dobré. Ostatné operácie sú na tom horšie, ak je na vrchu krabičky veľa mŕtvych prvkov. Niekedy to môže trvať až $O(n)$. Avšak priemerne na jednu operáciu to je oveľa lepšie. Sledujme život jedného prvku. Najprv ho vložíme do $MinKrab$ a $MaxKrab$. Potom ho vyberieme (bez ujmy na všeobecnosti) z $MinKrab$ a vložíme do $MaxMrtve$. Nakoniec Vyberieme z $MaxMrtve$ a $MaxKrab$. V priemere na jednu operáciu nám vychádza zložitosť $O(\log n)$, čo znie príjemnejšie. A čo pamäť? V krabičkách zaberieme priestor $2N$, lebo jeden prvok je naraz max. v 2 krabičkách. V samotnom programe používame premenné len na dočasné uloženie vyberaných údajov. Program by fungoval pre ľubovoľne veľké n, ak by sa nezaplnili čierne krabičky, teda $O(1)$. Podotýkam, že keby sme si v nejakej premennej pamätali počet prvkov v pamäti, potrebovali by sme $O(\log n)$ pamäte.

Listing programu:

```
var min_krab, max_krab,
    min_mrtve, max_mrtve :blackbox2;

procedure MMVyhodMrtve(krab, mrtve :blackbox2);
var a,b:integer;
begin
  repeat
    if Empty(krab) or Empty(mrtve) then
      exit;
    a:=ExtractMin(krab);
    b:=ExtractMin(mrtve);
  until a<>b;           {je mrtvy?}
  Insert(krab, a);
  Insert(mrtve, b);
end; {MMVyhodMrtve - Odstrani mrtve z vrchu krab}

function MMEmpty:boolean;
begin
  MMVyhodMrtve(min_krab, min_mrtve); {Nech ziju funkcie}
  MMVyhodMrtve(max_krab, max_mrtve); {s vedlajsim efektom}
  MMEmpty:=Empty(min_krab) and Empty(max_krab);
end; {Min/Max MMEmpty - Prazdna pamat? - Vyhodenie mrtvych}

procedure MMInsert(x:integer);
```

[1]0 cudzím slovom analogicky

```
begin
  Insert(min_krab,x);
  Insert(max_krab,-x);
end; {Min/Max Insert}

function MMExtract(a_krab, b_mrtve :blackbox2):integer;
var a:integer
begin
  if MMEmpty then       {Kde nic nie je, ani cert neberie}
    exit;               {Vyhodilo aj mrtve prvky}
  a:=ExtractMin(a_krab);
  MMExtract:=a;
  Insert(b_mrtve, -a);  {Umrel}
end; {Min/Max Extract - Vyberie zo zvolenej krabicky}

function MMExtractMin:integer;
begin
  MMExtractMin=MMExtract(min_krab, max_mrtve);
end; {Min/Max ExtractMin}

function MMExtractMax:integer;
begin
  MMExtractMin=-MMExtract(max_krab, max_mrtve);
end; {Min/Max ExtractMax}
```

Výsledková listina po 2. sérii kategórie KSP

	Meno a priezvisko	kola	Trieda		21	22	23	24	25	Σ
1.	Pavol Juhos	Gym. Grösslingová BA	4	71	13	15	15	15	14	143
	Peter Bella	Gym. Jura Hronca BA	4	73	15	15	15	11	14	143
3.	Radovan Bauer	Gym. Poštová Košice	4	66	14	15	15	9	9	128
4.	Pavol Mravec	Gym. K. Štúra Modra	4	67	15	13	8	2	14	119
5.	Ján Katrenič	Gym. Školská Spiš. Nová Ves	4	71	7	10	8	6	13	115
6.	Jakub Tekeľ	Gym. Jura Hronca BA	2	61	13	11	8	5	14	112
	Michal Malý	Gym. Žiar nad Hronom		70	7		15	10	10	112
8.	Jozef Tvarožek	Gym. Jura Hronca BA	4	44	14	15	15	9	14	111
9.	Ján Mazák	Gym. Poštová Košice	4	51	14	14	12	8	9	108
10.	Rasťo Šrámek	Gym. Tilgnerova BA	4	40	10	11	14	15	14	104
11.	Marek Tesař	Gym. Haličská Lučenec	4	45	7	15	8	13	14	102
12.	Vladimír Repiský	Gym. Žiar nad Hronom	4	67	14	6	8	2	3	100
13.	Pavol Müller	Gym. Grösslingová BA	4	51	9	1	8	11	14	94
14.	Michal Mikuš	Gym. Jura Hronca BA	4	54	7		8	10	9	88
15.	Juliana Lipková	Gym. Jura Hronca BA	3	35	7	14	8	10	13	87
	Peter Glaus	Gym. Jura Hronca BA	3	51	9		8	5	14	87
	Peter Truchlý	Gym. Párovská Nitra	4	60	6		7		14	87
18.	Monika Steinová	Gym. Einsteinova BA	4	51	8		8	10	9	86
19.	Marek Ludha	Gym. Tajovského B. Bystrica	2	48	9		8	11	8	84
20.	Peter Libič	Gym. Ľudovíta Štúra Trenčín	3	49	9	7	8		9	82
21.	Tomáš Dzetkulič	Gym. P. Horova Michalovce	4	48	9		8		14	79
22.	Anton Štefanek	Gym. Jura Hronca BA	2	45	7	7	7	3	9	78
23.	Tomáš Sághy	SPŠE Nové Zámky	4	56	7		8	2	3	76
24.	Anna Hanulová	Gym. Jura Hronca BA	2	35	9	14	7		9	74
25.	Tomáš Záhorec	Gym. Jura Hronca BA	4	37	3		7	8	15	70
26.	Michal Tekeľ	Gym. Konštantínova Prešov	4	42	7		8		8	65
27.	Peter Macko	Gym. Liptovský Hrádok	1	30	8	1	7	7	9	62
28.	Ivan Šomlo	Gym. Mládežnícka Šahy	2	48	7	1	0	0	3	59
29.	Jaroslav Klíma	Gym. Jura Hronca BA	3	58						58
	Jozef Veselý	Gym. Jura Hronca BA	3	58						58
31.	Matúš Harvan	Gym. Jura Hronca BA	4	56						56
	Miroslav Cicko		1	51	5					56
	Oto Vozár	Gym. Palackého BA	3	56						56
34.	Juraj Andris	Gym. Sereď	4	29	9		7	8	2	55
35.	Tomáš Vrábel	Gym. Malá Hora Martin	4	54						54
36.	Juraj Sloboda			53						53
	Ladislav Vadkerti	SPŠE Piešťany		20	2	7	7	8	9	53
38.	Martin Choma	Gym. Stará Ľubovňa	3	24	3	10	6		9	52
	Radovan Čulák	Gym. Golianova Nitra	2	11	9		8	17	7	52
40.	Ľuboš Bednárik	Gym. Ľudovíta Štúra Trenčín	4	9	7		8	16	9	49
41.	Radovan Berta	Gym. Daxnera V.n. Topľou	3	34	6				8	48
42.	Ján Palenčár	Gym. Malá Hora Martin	2	28	3		2	5	8	46
43.	Ján Veselý	Gym. Golianova Nitra	2	19	7	2	6	4	7	45
	Oliver Janík	Gym. L. Stöckela Bardejov	2	21	9		7		8	45
45.	Marek Hanes		1	23	1		8	2	9	43
	Milan Šatka	Gym. Liptovský Hrádok	3	0	1	10	8	10	14	43
47.	Matej Max	Gym. Alejová Košice	4	40						40
48.	Milan Burda	Gym. Golianova Nitra	2	15	4	2	7	2	9	39
	Peter Dzurjanin	Gym. Grösslingová BA	4	39						39
50.	Lucia Tiererová	Gym. Jura Hronca BA	4	36						36

	Meno a priezvisko	kola	Trieda		21	22	23	24	25	Σ
	Štefan Konečný	Gym. Jura Hronca BA	3	12	14		7		3	36
52.	Michal Rjaško	Gym. Daxnera V.n. Topľou	3	0	8		8	8	9	33
53.	Jakub Kováč	Gym. Jura Hronca BA	2	0	7		7	8	10	32
	Marián Schmotzer	Gym. Horvátha BA	2	17	6				9	32
55.	Peter Greškovič	Gym. Svidník	1	17	7		7			31
56.	Pavol Rusnák	Gym. Konštantínova Prešov	4	27						27
57.	Filip Karas	Gym. Golianova Nitra	2	8	3		6	2	5	24
58.	Marek Tomacha	Gym. sv. Uršule BA	4	21						21
59.	Jana Šišková	Evanjelické gym. B. Bystrica	4	19						19
	Marek Cifra	Gym. A. Merici Trnava	2	19						19
61.	László Marák	Gym. H. Selyeho Komárno	3	0	5	1	7	2	3	18
62.	Peter Kopčanský	Gym. Krompachy	4	13			1			14
63.	Peter Šufliarsky		2	13						13
64.	Michal Čerešňa	Gym. Nové Zámky	2	12						12
65.	Ladislav Hodermarský	SPŠE Stará Turá	4	11						11
66.	Ján Hutár	SPŠE Piešťany	3	1						1
	Miroslav Hudák	Gym. Golianova Nitra	4	1						1