

### Soluția I – $N=K - 10p$

În acest caz, fiecare fecior va lua câte un sac din cei  $N$ . Vom împărți sacii astfel: feciorul 1 va lua sacul cu cei mai mulți galbeni, feciorul 2 pe următorul ș.a.m.d. până când feciorul cel mai mic ia sacul cu cei mai puțini galbeni. În caz de egalitate nu trebuie să facem nimic special pentru că un fecior mai mare poate primi aceeași sumă de galbeni ca un fecior mai mic.

Construim 3 vectori:

$a[i]$  - numărul de galbeni al fiecărui sac  $i$  în ordinea din fișierul de intrare;

$ind[i]$  - indicele sacului (inițial  $ind[i]=i$ );

$fiu[i]$  - indicele fiului ce va lua sacul respectiv (necompletat inițial).

Sortăm descrescător vectorul  $a$  modificând și vectorul  $ind$  corespunzător. Pe prima linie putem afișa valoarea celui mai mic sac,  $a[N]$ .

Apoi, vom completa vectorul  $fiu$  printr-o parcurgere de la stânga la dreapta:  $fiu[i]=i$ .

În acest moment un grup  $(a[i], ind[i], fiu[i])$  reprezintă nr de galbeni din sac, indicele inițial al sacului și indicele fiului ce va primi acel sac. Pentru a reconstrui împărțirea, sortăm crescător vectorul  $ind$  modificând și vectorul  $fiu$  corespunzător. Acum în vectorul  $fiu$  avem ordinea în care fii trebuie chemați feciorii de împart.

Pentru că avem de sortat vectorii, complexitatea timp a acestei soluții este  $O(N \cdot \log(N))$  sau  $O(N^2)$ , în funcție de algoritmul de sortare ales.

### Soluția II – “brute force” – 30p

Vom construi toate variantele posibile de a împărți sacii. Observăm că trebuie să generăm  $K-1$  poziții din cele  $N$ , fie ele  $s[1], s[2], \dots, s[K-1]$ .

Astfel, un fiu va primi sacii cu indicii între 1 și  $s[1]$ , alt fiu va primi sacii cu indicii de la  $s[1]+1$  la  $s[2]$ , ..., ultimul fiu va primi sacii cu indicii de la  $s[K-1]+1$  la  $N$ .

Generăm cu un algoritm de tip succesor toate variantele posibile. Vom păstra soluția în care fiul cel mic ia cei mai mult posibil.

Pentru fiecare împărțire posibilă, calculăm cât ia fiecare fiu și ordinea în care feciorii sunt chemați folosind un raționament similar cu cel de la Soluția I: calculăm toate cele  $K$  sume de galbeni într-un vector pe care îl sortăm descrescător.

Atribuirem fiului cu indicele 1 cea mai mare sumă de galbeni, apoi fiului 2 următoarea cea mai mare sumă de galbeni (NU uităm că poate fi egală cu cea precedentă), ș.a.m.d. Mezinul va lua cea mai mică sumă de galbeni. Pentru afișarea împărțirii este necesară încă o sortare a fiilor după indicii sacilor luați.

Complexitatea acestei soluții este exponențială,  $O(2^N)$ . O soluție de această complexitate ar trebui să primească cel puțin 30 de puncte.

### Soluția III – $O(N^3)$ – 60p

Se observă că dacă cel mai mic dintre feciori ia o anumită sumă de galbeni  $x$ , atunci toți feciorii mai mari ca el trebuie să ia cel puțin  $x$  galbeni.

Cum fiecare fecior va primi o subsecvență continuă de saci, ne putem fixa secvența de saci pe care o va primi cel mai mic fecior.

Fie această secvență  $[i, j]$  cu suma de galbeni  $X$ . Trebuie să verificăm dacă putem împărți sacii  $[1, i-1]$  și sacii  $[j+1, N]$  în **cel puțin**  $K-1$  subsecvențe continue cu suma mai mare ca  $X$ .

Pentru aceasta, vom face două parcurgeri: una pornind din  $i$  spre stânga, cealaltă pornind din  $j$  spre dreapta. Vom reține la fiecare pas o variabilă  $suma$  și un contor  $cnt$ , inițial egale cu 0. Pentru fiecare sac în ordinea parcurgerii, vom aduna numărul de galbeni la suma curentă. Când aceasta va deveni mai mare sau egală cu  $X$ , vom reseta  $suma$  și vom incrementa  $cnt$ . Făcând același lucru și pentru partea din dreapta, putem verifica dacă subsecvența  $[i, j]$  este validă, verificând ca suma celor două contoare să depășească  $K-1$ .

Dintre toate subsecvențele valide, o vom păstra pe cea cu suma cea mai mare.

Această soluție are complexitate  $O(N^3)$  și ar trebui să obțină cel puțin 60 de puncte.

## Reconstrucția împărțirii

Se poate adapta soluția de mai sus pentru a afișa și cele  $K$  subsecvențe alese, în cazurile valide. Pentru a asigura cele  $K$  subsecvențe celor  $K$  fii, trebuie să ne asigurăm că averile vor fi atribuite fiilor în ordinea vârstei sale. Acest lucru se va realiza prin doi pași:

**Pasul 1:** Aflăm fiecare subsecvență cărui fiu îi aparține. Pentru aceasta, atribuim fiecărei subsecvențe numere de la 1 la  $K$ , în ordinea descrescătoare a numărului de galbeni. Acest lucru se poate realiza printr-o sortare.

**Pasul 2:** Afișăm lungimea subsecvenței și fiul corespunzător, în ordinea de la stânga la dreapta.

O atenție specială trebuie acordată cazului în care soluția de mai sus va afișa mai mult de  $K$  subsecvențe, caz în care unele subsecvențe vecine vor trebui "contopite". Din fericire, însă, se poate observa că putem alege să "contopim" oricare două subsecvențe vecine, soluția rămânând validă.

Complexitatea reconstrucției este  $O(N^2)$  sau  $O(N \cdot \log(N))$ , în funcție de algoritmul de sortare ales.

## Soluția IV - $O(N \cdot \log(\text{suma\_totala\_galbeni}))$ - 90p

Putem îmbunătăți soluția precedentă **căutând binar** cel mai mare  $X$  între  $[1, S]$  (unde  $S$  este suma totală de galbeni din saci), astfel încât să existe  $K$  subsecvențe în șirul de saci, cu suma fiecăreia cel puțin egală cu  $X$ . Pentru un  $X$  fixat vom verifica dacă există  $K$  astfel de subsecvențe. Dacă nu există, atunci vom căuta printre valori mai mici.

Pentru a verifica dacă un  $X$  fixat este "bun", vom proceda ca și în soluția precedentă: pornim cu o variabilă  $suma$  și un contor  $cnt$ , inițial egale cu 0. Parcurgem sacii de la 1 la  $N$  și adunăm pe rând numărul de galbeni din sacul  $i$  la suma curentă. Când suma devine mai mare sau egală cu  $X$  atunci mărim contorul  $cnt$  cu 1 și resetăm suma. Dacă, la final  $cnt$  devine cel puțin egal cu  $K$  atunci  $X$ -ul fixat este "bun". Diferența este că, în acest caz, nu mai trebuie să ne fixăm subsecvența de sumă egală cu  $X$ .

Căutarea binară va face  $\log(S)$  pași, iar la fiecare pas avem o parcurgere liniară pentru verificare. Reconstituirea se realizează similar cu soluția precedentă.

Complexitatea finală este  $O(N \cdot (\log(S) + \log(N)))$ . O soluție cu o complexitate similară ar trebui să primească 90 de puncte.