Matematică şi algoritmi

Irina MUSTAŢĂ[1]

Majoritatea problemelor de informatică se rezolvă cu ajutorul unei serii de algoritmi, care se adaptează de la caz la caz. Din problemele de mai jos, se va vedea că uneori anumite observaţii matematice fie uşurează abordarea acestor probleme, fie întăresc rezultatul.

1. La **Concursul Info-Oltenia** din 2004 s-a dat următoarea problemă:

Să se afişeze numărul de descompuneri ale unui număr dat în sumă de numere naturale nenule. O descompunere şi o permutare a ei nu se vor număra de două ori. (Enunţ adaptat).

Un programator "conştiincios" va aplica probabil backtracking (algoritm care generează toate soluţiile de descompunere) şi apoi le va contoriza. Deşi o astfel de abordare este corectă, se ştie că backtracking-ul are o complexitate (număr de operaţii efectuate) exponenţială, determinând un timp de execuţie mai îndelungat.

Să privim problema altfel: vom încerca să găsim o funcţie recursivă care returnează numărul căutat. Pentru a evita numărarea simultană a unei soluţii şi a permutării ei, vom lua numerele care apar în descompunerea lui n în ordine crescătoare. Presupunem că putem construi o funcţie recursivă $f(n)$ (cu un singur parametru) care returnează numărul cerut. Atunci $f(n) = f(n-1) + f(n-2) + \cdots + f(1)$. (Adică din n scădem k iar numărul de descompuneri posibile va fi $f(n-k)$, apoi facem suma după k). Numai că, în acest caz apare o problemă majoră: dacă scriem $n = 2+n-2$ şi calculăm $f(n-2)$, la un moment dat primul termen al descompunerii ar fi 1, deci am obţine $n = 2 + 1 + \ldots$, ceea ce contrazice ordonarea aleasă pentru numerele din descompunerea lui n.

Se impune, aşadar, necesitatea construirii unei funcţii cu doi parametri $f(n,k)$, cu următoarea semnificaţie: $f(n,k)$ reprezintă numărul de descompuneri distincte ale lui n în sumă de termeni $\geq k$. Este clar că enunţul problemei cere $f(n,1)$. De asemenea se observă că $f(n,n) = 1$ şi $f(n,k) = 0$ pentru $k > n$.

Acum putem scrie definiţia funcţiei $f(n,k)$, căreia îi putem da ulterior parametrii n, 1:

$$f(n,k) = \begin{cases} 1, & n = k, \\ 0, & n < k, \\ \sum\limits_{l=k}^{n-1} f(n-l,l), & n > k. \end{cases}$$

Se observă că atât codul, cât şi resursele consumate sunt mult mai restrânse decât în cazul backtracking-ului.

2. O altă problemă de acelaşi gen ar fi:

Afişaţi toate soluţiile ecuaţiei $3x + y + 4xz = 100$ *cu* $x, y, z \in \mathbb{N}$.

[1] Elevă, cl. a XII-a, Colegiul Naţional, Iaşi

Şi în acest caz este de aşteptat o aplicare de backtracking, dar putem reformula ecuaţia: $x\,(4z+3) = 100 - y$. Acum raţionăm în felul următor. Alegem z de la 0 la 24 şi, pentru z astfel luat, alegem x de la 1 la $\left[\dfrac{100}{4z+3}\right]$. Apoi îl obţinem pe y din $100 - x\,(4z+3)$ şi tipărim tripletul (x, y, z). În final tipărim şi soluţiile $(0, 100, z)$ cu $z \in \mathbb{N}$. Prin metoda de mai sus, obţinem într-adevăr toate soluţiile (z poate lua orice valoare de la 0 la 24, deoarece pentru $z \geq 25$ şi x nenul avem $4z + 3 > 100$; totodată $100 - y \leq 100$, deci $x \leq \dfrac{100}{4z+3}$ şi, cum $x \in \mathbb{N}$, $x \leq \left[\dfrac{100}{4z+3}\right]$).

Observaţie. Numărul 100 poate fi înlocuit cu orice $n \in \mathbb{N}^*$.

3. Un alt exerciţiu interesant este următorul:

Se dau m numere întregi nenule $b_1, \ldots, b_m$ şi n numere întregi nenule $a_1, a_2, \ldots, a_n$. Să se determine o submulţime a mulţimi $B = \{b_1, \ldots, b_m\}$ care să maximizeze valoarea expresiei

$$E = a_1x_1 + a_2x_2 + \cdots + a_nx_n,$$

ştiind că $m \geq n$ şi $x_i \in \{b_1, \ldots, b_m\}$, $i = \overline{1,n}$.

Sortăm crescător ambele şiruri (a_i) şi (b_i). Evident, la ambele şiruri vom avea numerele negative în partea stângă şi cele pozitive în partea dreaptă.

Vom enunţa *inegalitatea rearanjărilor*, pe care o vom aplica la această problemă:

Fie $a_1 \leq a_2 \leq \cdots \leq a_n$, $b_1 \leq b_2 \leq \cdots \leq b_n$ şiruri de numere reale şi $S(\sigma) = \sum_{i=1}^{n} a_i b_{\sigma(i)}$, unde σ este o permutare a mulţimii $\{1, 2, \ldots, n\}$. Atunci $S(\sigma)$ este maxim pentru $\sigma(i) = i$, $\forall i = \overline{1,n}$, şi minim pentru $\sigma(i) = n + 1 - i$, $\forall i \in \overline{1,n}$.

Şirul (a_i) este alcătuit din k numere strict negative şi $n - k$ numere întregi strict pozitive. Pentru a maximiza suma este necesar să înmulţim numerele pozitive din (a_i) cu cele mai mari numere din (b_i) şi numerele negative cu cele mai mici numere din (b_i) (deoarece cele mai mici numere strict negative din (b_i) au modulul mai mare, deci $a_ib_i > 0$ va putea fi maximizat, iar cele mai mici numere pozitive înmulţite cu numere negative vor da cele mai mici în modul numere negative, deci cei mai mari termeni). Aşadar vom înmulţi k numere negative ale şirului (a_i) cu cele mai mici k numere ale şirului (b_i), iar cele $n - k$ numere pozitive ale şirului (a_i) le vom înmulţi cu cele mai mari $n - k$ numere ale şirului (b_i). Aceste două submulţimi nu se vor suprapune, întrucât $m \geq n$, iar numărătoarea începe din capetele opuse ale şirului (b_i) sortat. Din inegalitatea rearanjărilor obţinem că permutarea lui $\{b_1, \ldots, b_k\}$ pentru care $\sum_{i=1}^{k} a_ib_i$ este maximă e permutarea identică şi analog pentru mulţimile $\{a_{k+1}, \ldots, a_n\}$ şi $\{b_{m-n+k+1}, \ldots, b_m\}$.

Să remarcăm că programul propriu-zis va fi astfel redus la nişte paşi extrem de simpli: sortarea crescătoare a şirurilor (a_i) şi (b_i), aflarea numărului k de elemente negative ale lui (a_i) şi, în final, tipărirea soluţiei: primele k numere din stânga şirului (b_i) şi ultimele $n - k$ numere din dreapta din (b_i).

4. *Un colier este format din mărgele roşii şi albastre. La un moment dat tăiem colierul într-un punct şi îl întindem în linie dreaptă. Apoi scoatem de pe fir mărgelele din capătul stâng până când întâlnim una din cealaltă culoare. Analog procedăm pentru capătul drept. Dându-se datele despre colier sub forma unui şir rarra... (care indică culoarea mărgelelor succesive), să se calculeze numărul maxim de mărgele pe care îl putem scoate, precum şi locul unde trebuie tăiat colierul.* (Enunţ adaptat).

(**Olimpiada Internaţională de Informatică**, 1993)

Considerăm un vector cu n componente (adică un şir $a_1, \ldots, a_n$). Iniţializăm cu 0 toate elementele vectorului, ne plasăm pe primul element al vectorului şi citim şirul colierului caracter cu caracter, câtă vreme caracterul citit este acelaşi cu cel anterior; mărim cu 1 valoarea reţinută de primul element al vectorului. Când caracterul se schimbă ne poziţionăm pe următoarea componentă a vectorului şi continuăm până la sfârşitul şirului cu acelaşi procedeu. De fapt, vectorul reţine lungimile tuturor secvenţelor succesive de mărgele colorate la fel.

Când tăiem colierul, o putem face în două feluri: fie în interiorul unei secvenţe colorate la fel, fie despărţind două secvenţe colorate diferit. În primul caz, după ce întindem colierul, numărul maxim de mărgele ce pot fi scoase este numărul de mărgele al acelei secvenţe, în timp ce în al doilea caz, numărul maxim obţinut este suma numerelor de mărgele din două secvenţe consecutive. Este evident că pentru noi e mai convenabil să tăiem colierul între două secvenţe.

Maximul care poate fi obţinut este maximul dintre sumele numerelor de mărgele a două secvenţe succesive. Acesta se poate găsi uitându-ne la vectorul completat anterior. Fie k numărul de elemente nenule al acestui vector. Dacă k este par, comparăm sumele $v[1]+v[2]$, $v[2]+v[3]$, $\ldots$, $v[k-1]+v[k]$. Dacă k este impar, cum oricare două secvenţe consecutive au culori diferite, rezultă că secvenţele numărate de $v[1]$ şi $v[k]$ au aceeaşi culoare, deci la "închiderea" colierului se vor uni într-o singură secvenţă de lungime $v[1] + v[k]$. În acest caz, trebuie aflat maximul dintre $v[k] + v[1] + v[2]$, $v[1] + v[k] + v[k-1]$, $v[i] + v[i+1]$, $i = \overline{2, k-2}$.

Locul unde trebuie tăiat colierul se obţine uşor. Presupunem că facem tăietura între $v[i]$ şi $v[i+1]$. Atunci poziţia unde vom tăia este $\sum_{j=1}^{i} v[j]$.

Recreaţii ... matematice

Soluţiile problemelor enunţate la p. 23.

1. Mutând un beţişor de la numărător deasupra numărului din membrul doi, se obţine

$$\frac{XXII}{VII} = II,$$

care este o egalitate aproximativă cu o eroare foarte mică ($\frac{XXII}{VII} = \frac{22}{7} \approx 3,14285\ldots$, iar $\pi \approx 3,14159\ldots$).

2. Se adaugă cifra 0, dar ... la exponent, obţinându-se numărul impar 1 (de exemplu, $7^0 = 1$ etc.).