

Programarea Algoritmilor

– LABORATOR NR. 6 –

Divide et Impera, Backtracking

I. Divide et Impera

1. (Problema rucsacului)

Se dă o mulțime formată din N obiecte, fiecare fiind caracterizat de o greutate și un profit. Să se găsească o submulțime de obiecte astfel încât suma profiturilor lor să fie maximă, iar suma greutăților lor să nu depășească o valoare G. În fișierul “rucsac.in” se găsește pe prima linie greutatea maximă admisă în ghiozdan urmată de mai multe linii ce conțin greutatea și câștigul produsului respectiv.

Indicație de rezolvare:

- (seria 13, curs 11: quickselect, algoritmul BFPRT)

Exemplu:

rucsac.in	rucsac.out	Explicație
50	7 5.0 50.0 10.0 1	Fiecare linie din afișare conține: - id-ul obiectului (al câtelea era în fișierul de intrare) - greutatea - câștigul - raportul câștig/greutate - procentul din obiect care a fost pus în rucsac
21 63	4 25.0 100.0 4.0 1	
10 10	8 10.0 40.0 4.0 1	
10 30	9 5.0 20.0 4.0 1	
25 100	1 21.0 63.0 3.0 0.238	
38 19	Castig maxim: 225.0	
7 14		
5 50		
10 40		
5 20		

2. (Zona dreptunghiulară cu arie maximă)

Într-o zonă rezidențială se află o pădure foarte frumoasă, de forma unui dreptunghi. Un investitor isteț s-a gândit să-și construiască o vilă chiar în pădure, dar, fiind un ecologist convins, nu ar vrea să taie niciun copac. Din acest motiv, el ar vrea să afle zona dreptunghiulară din pădure cu suprafață maximă și în care nu este niciun copac. Investitorul are o hartă a întregii zone, în care sunt date coordonatele dreptunghiului corespunzător pădurii, precum și coordonatele tuturor copacilor din ea.

Indicație de rezolvare:

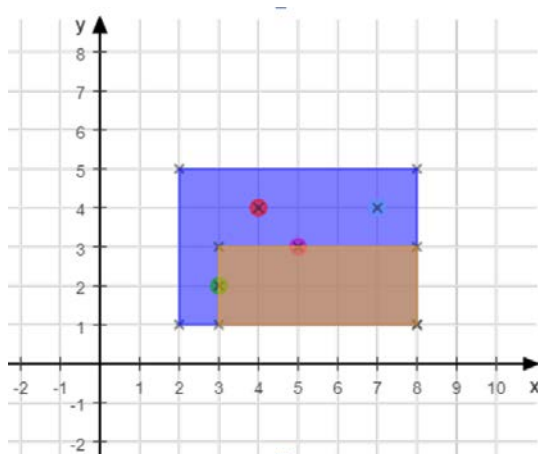
- Se folosește un tuplu pentru coordonatele dreptunghiului de arie maximă, o listă de tupluri pentru valorile coordonatelor citite din fișier și o variabilă pentru aria maximă.
- Se creează o funcție “citireDate()” ce întoarce: primele 4 valori pentru a obține coordonatele pentru colțul stânga-jos și dreapta-sus și o listă de tupluri pentru restul punctelor.
- Se creează o funcție “dreptunghiArieMaxima(xst, yst, xdr, ydr)” ce primește ca parametri inițiali colțul stânga-jos și dreapta-sus al dreptunghiului total (pădurea întreagă).
 - Pentru fiecare copac (din lista de tupluri), se verifică dacă este inclus strict în interiorul dreptunghiului curent (determinat de cei 4 parametri ai funcției): dacă da, se “taie” întreg dreptunghiul pe verticală la coordonata x a copacului (și se fac 2 apeluri recursive ale funcției,

pentru dreptunghiul din stânga și cel din dreapta copacului), apoi se “taie” pe orizontală la coordonata y a copacului (și se fac 2 apeluri recursive ale funcției, pentru dreptunghiul de deasupra și cel de dedesubtul copacului).

- Dacă dreptunghiul curent nu conține niciun copac, atunci i se calculează aria și dacă e mai mare decât aria maximă curentă, atunci se actualizează aria maximă, precum și coordonatele dreptunghiului de arie maximă.

Exemplu:

copaci.in	copaci.out	Explicație
2 1	Dreptunghiul:	Pădurea este un dreptunghi având colțul stânga-jos de coordonate (2,1) și colțul dreapta-sus de coordonate (8,5). În pădure sunt 4 copaci, având coordonatele (3,2), (4,4), (5,3) și (7,4).
8 5	3 1	
3 2	8 3	Dreptunghiul cu suprafața maximă de 10 și care nu conține nici un copac are coordonatele (3,1) pentru colțul stânga-jos și (8,3) pentru colțul dreapta-sus.
4 4	Aria maxima:	
5 3	10	
7 4		



3. (Numărul de apariții ale unui element într-o listă sortată)

În fișierul “data.in” se află pe prima linie elementele unei liste, iar pe a doua linie se afla un element x. Sortați lista și determinați numărul de apariții al unui element x din listă folosind căutarea binară.

Indicație de rezolvare:

- Se realizează o funcție “citireDate()” ce întoarce o listă formată din elementele aflate pe prima linie și o variabilă.
- Se realizează o funcție “contor(lista, x, n)” , unde lista este cea formată din valorile citite din fișier sortată, x elementul ce trebuie căutat, n numărul de elemente ale listei. Aceasta apelează funcțiile primaAparitie și ultimaAparitie și returnează diferența dintre cele două poziții returnate de funcții.

- Se realizează o funcție “primaAparitie(lista, pozMinima, pozMaxima, x, n)” care calculează mijlocul intervalului din pozMinima și pozMaxima și verifică dacă x se poate afla între cele două poziții transmise ca parametru returnând poziția, altfel returnează -1.
- Se realizează o funcție “ultimaAparitie(lista, pozMinima, pozMaxima, x, n)” ce lucrează în același mod ca și funcția definită anterior, dar întoarce valoarea ultimei apariții a lui x.

Exemplu:

data.in	data.out
2 1 3 2 3 3	3
3	

4. Se dă un vector $a=(a_1,...,a_n)$ de tip munte (există un indice i astfel încât $a_1 < a_2 < ... < a_i > a_{i+1} > ... > a_n$; a_i se numește vârful muntelui). Propuneți un algoritm $O(\log n)$ care determină vârful muntelui (în calculul complexității algoritmului nu se consideră și citirea vectorului).

Indiciu de rezolvare: gândire asemănătoare ca în cazul căutării binare.

date.in	date.out
5	11
4 8 10 11 5	

5. a) Se citește de la tastatură un număr natural N . Se consideră o tablă (matrice) pătratică de dimensiuni $2^N \times 2^N$ pe care se scriu numerele naturale de la 1 și $2^N \times 2^N$ prin vizitarea **recursivă** a celor patru cadrane ale tablei în ordinea indicată și în figura alăturată: dreapta-sus, stânga-jos, stânga-sus, dreapta-jos. De exemplu, dacă $N=2$, tabla este completată astfel:

```

11  9  3  1
10 12  2  4
 7  5 15 13
 6  8 14 16


```

Să se afișeze în fișierul tabla.out matricea completată după regulile precizate.

intrare	tabla.out
2	11 9 3 1 10 12 2 4 7 5 15 13 6 8 14 16

Indicație de rezolvare: Dacă matricea este de dimensiune 1×1 , se completează valoarea corespunzătoare. Altfel se împarte matricea în 4 cadrane și se apelează recursiv pentru fiecare cadran, în ordine, și cu valoarea de la care trebuie să înceapă completarea cadranelor.

- b) <http://www.infoarena.ro/problema/z> (fără a memora tabla) $O(kn)$

6. Fie o tablă cu pătrățele de dimensiune $2^n \times 2^n$ (n dat). Pe această tablă există o gaură la o poziție dată prin linia și coloana sa (lg, cg) (liniile și coloanele se consideră numerotate de la 1). Pentru acoperirea acestei table avem la dispoziție piese de forma 

Aceste piese pot fi rotite cu 90° , 180° sau 270° . Să se afișeze o acoperire completă a tablei (cu excepția găurii). Piesele vor fi reprezentate prin numere de la 1 la n , iar gaura prin 0 (cele 3 căsuțe ocupate de a i-a piesă pusă vor primi valoarea i) $O(2^{2n})$

date.in	date.out (un exemplu, soluția nu este unică)
2 3 1	1 1 2 2 1 3 3 2 0 4 3 5 4 4 5 5

Indicație de rezolvare: Matricea se împarte în 4 cadrane. Unul dintre acestea va conține "gaura", 3 dintre acestea nu. Se plasează o piesă în forma de L în centrul figurii, astfel încât cate un câmp din piesă se va afla în fiecare dintre cele 3 cadrane. Se apelează funcția recursiv pentru fiecare cadran.

7. Se dau doi vectori a și b de lungime n , respectiv m , cu elementele ordonate crescător. Propuneți un algoritm cât mai eficient pentru a determina mediana vectorului obținut prin interclasarea celor doi vectori $O(\log(\min\{n,m\}))$

(<https://leetcode.com/problems/median-of-two-sorted-arrays/> + curs seria 14)

date.in	date.out
10 4 6 8 10 12 14 16 18 20 3 1 23 25	12

Indicații rezolvare: curs/seminar

II. Backtracking

1. (Descompunere)

În fișierul "descompunere.in" se află un număr. Descompuneți acest număr ca o sumă de numere naturale distincte. Afișează rezolvarea în "descompunere.out".

Indicație de rezolvare:

- se realizează o funcție "bkt(k)" ce va pune pe poziția k dintr-o listă de soluții valorile din intervalul $[1; n-k+2]$, realizează suma, dacă aceasta este mai mică sau egală cu n, se verifică dacă aceasta este n și se afișează elementele din lista de soluții până la poziția k, altfel se apelează funcția cu următoarea valoare a lui k;

Exemplu:

descompunere.in	descompunere.out
4	1 1 1 1 1 1 2 1 2 1 1 3 2 1 1 2 2 3 1 4

- pentru descompuneri distincte, respectiv eliminarea descompunerilor formate din aceiași termeni este necesar să populăm lista de soluții cu valori din intervalul $[1; n-k+2]$ dacă valoarea lui k este 0, altfel din intervalul $[sol[k-1]; n-k+2]$;

Exemplu:

descompunere.in	descompunere.out
4	1 1 1 1 1 1 2 1 3 2 2 4

- pentru descompuneri distincte cu termeni distincți este necesar să populăm lista de soluții cu valori din intervalul $[1; n-k+2]$ dacă valoarea lui k este 0, altfel din intervalul $[sol[k-1]+1; n-k+2]$.

Exemplu:

descompunere.in	descompunere.out
4	1 3 4

2. Generare

În fișierul "generare.in" se află un număr s . Generați toate numerele cu cifre distincte și suma s . Afișează aceste valori în fișierul "generare.out".

Indicație de rezolvare:

- se realizează o funcție "bkt(k)" ce va pune pe poziția k dintr-o listă de soluții valorile din intervalul $[1; 10]$ dacă k este 0, altfel $[0; 10]$, realizează suma, dacă aceasta este mai mică sau egală cu n se verifică dacă aceasta este n și se afișează elementele din lista de soluții până la poziția k , altfel se apelează funcția cu următoarea valoare a lui k .
- Pentru fiecare soluție, se afișează ea, iar dacă nu conține cifra 0, atunci se afișează și ea cu 0 adăugat la final (ex: 12 și 120).

Exemplu:

generare.in	generare.out
3	102 12 120 201 21 210 3 30

3. Problema colorării hărților

În fișierul "harti.in" se află pe prima linie numărul natural n , urmat de valori i și j reprezentând 2 țări cu frontieră comună pe linii diferite. Afișează toate modalitățile în care pot fi colorate hărțile astfel încât 2 țări cu graniță comună să nu fie colorate identic.

Indicație de rezolvare:

- se realizează un dicționar pentru care cheia este numărul țării iar valoarea este o listă compusă din toți vecinii țării respective;
- se realizează o funcție “bkt(k)” în care se generează toate culorile din intervalul [1; 4] și se verifică dacă țara curentă are un vecin de aceeași culoare.

Exemplu:

harti.in	afisare
6	[1, 2, 3, 2, 3, 4] [1, 2, 3, 2, 4, 4] [1, 2, 3, 4, 3, 2] [1, 2, 4, 2, 3, 3]
1 2	[1, 2, 4, 2, 4, 3] [1, 2, 4, 3, 4, 2] [1, 3, 2, 3, 2, 4] [1, 3, 2, 3, 4, 4]
3 1	[1, 3, 2, 4, 2, 3] [1, 3, 4, 2, 4, 3] [1, 3, 4, 3, 2, 2] [1, 3, 4, 3, 4, 2]
4 1	[1, 4, 2, 3, 2, 4] [1, 4, 2, 4, 2, 3] [1, 4, 2, 4, 3, 3] [1, 4, 3, 2, 3, 4]
1 5	[1, 4, 3, 4, 2, 2] [1, 4, 3, 4, 3, 2] [2, 1, 3, 1, 3, 4] [2, 1, 3, 1, 4, 4]
2 3	[2, 1, 3, 4, 3, 1] [2, 1, 4, 1, 3, 3] [2, 1, 4, 1, 4, 3] [2, 1, 4, 3, 4, 1]
5 2	[2, 3, 1, 3, 1, 4] [2, 3, 1, 3, 4, 4] [2, 3, 1, 4, 1, 3] [2, 3, 4, 1, 4, 3]
4 3	[2, 3, 4, 3, 1, 1] [2, 3, 4, 3, 4, 1] [2, 4, 1, 3, 1, 4] [2, 4, 1, 4, 1, 3]
5 4	[2, 4, 1, 4, 3, 3] [2, 4, 3, 1, 3, 4] [2, 4, 3, 4, 1, 1] [2, 4, 3, 4, 3, 1]
6 1	[3, 1, 2, 1, 2, 4] [3, 1, 2, 1, 4, 4] [3, 1, 2, 4, 2, 1] [3, 1, 4, 1, 2, 2]
3 6	[3, 1, 4, 1, 4, 2] [3, 1, 4, 2, 4, 1] [3, 2, 1, 2, 1, 4] [3, 2, 1, 2, 4, 4]
4 6	[3, 2, 1, 4, 1, 2] [3, 2, 4, 1, 4, 2] [3, 2, 4, 2, 1, 1] [3, 2, 4, 2, 4, 1]
	[3, 4, 1, 2, 1, 4] [3, 4, 1, 4, 1, 2] [3, 4, 1, 4, 2, 2] [3, 4, 2, 1, 2, 4]
	[3, 4, 2, 4, 1, 1] [3, 4, 2, 4, 2, 1] [4, 1, 2, 1, 2, 3] [4, 1, 2, 1, 3, 3]
	[4, 1, 2, 3, 2, 1] [4, 1, 3, 1, 2, 2] [4, 1, 3, 1, 3, 2] [4, 1, 3, 2, 3, 1]
	[4, 2, 1, 2, 1, 3] [4, 2, 1, 2, 3, 3] [4, 2, 1, 3, 1, 2] [4, 2, 3, 1, 3, 2]
	[4, 2, 3, 2, 1, 1] [4, 2, 3, 2, 3, 1] [4, 3, 1, 2, 1, 3] [4, 3, 1, 3, 1, 2]
	[4, 3, 1, 3, 2, 2] [4, 3, 2, 1, 2, 3] [4, 3, 2, 3, 1, 1] [4, 3, 2, 3, 2, 1]

4. Programarea spectacolelor

În fișierul "spectacole.txt" se află ora de început, ora de final și numele unor spectacole. Realizați un program care afișeze toate modalitățile de programare a unui număr maxim de spectacole care să nu se suprapună într-o sală dată.

Exemplu:

spectacole.txt	programare.txt
10:00-11:20 Scufita Rosie 09:30-12:10 Punguta cu doi bani 08:20-09:50 Vrajitorul din Oz 11:30-14:00 Capra cu trei iezi 12:10-13:10 Micul Print 14:00-16:00 Povestea porcului 15:00-15:30 Frumoasa din padurea adormita	08:20-09:50 Vrajitorul din Oz 10:00-11:20 Scufita Rosie 12:10-13:10 Micul Print 15:00-15:30 Frumoasa din padurea adormita 08:20-09:50 Vrajitorul din Oz 10:00-11:20 Scufita Rosie 12:10-13:10 Micul Print 14:00-16:00 Povestea porcului 08:20-09:50 Vrajitorul din Oz 10:00-11:20 Scufita Rosie 11:30-14:00 Capra cu trei iezi 15:00-15:30 Frumoasa din padurea adormita 08:20-09:50 Vrajitorul din Oz 10:00-11:20 Scufita Rosie 11:30-14:00 Capra cu trei iezi 14:00-16:00 Povestea porcului