Tehnici de programare

Greedy

ovidiu.banias@aut.upt.ro

Greedy

- ***Submulțime maximală***
- ***Planificarea activităților***
- ***Problema rucsacului***
- ***Greedy euristic***
- ***Codificare Huffman***

Greedy

- Se pretează problemelor decizionale de genul: se dă o mulțime A, să se găsească o submulțime B care îndeplineşte anumite condiții
- Se alege “optimul local” la fiecare pas pentru a obține “optimul global”
- Greedy (soluție optimă) vs Greedy euristic (soluție aproape de optim/soluție acceptabilă)

Greedy. Formalizare

$$A = \{x_1, x_2, x_3, ..., x_n\}$$

$$D_1 \in A - \text{optim local}$$

$$D_2 \in A - \text{optim local}$$

.

.

$$D_i \in A - \text{optim local}$$

.

.

$$D_k \in A - \text{optim local}$$

$$B = \{D_1, D_2, D_3, ... D_k) / D_i - \text{optim local}$$

$$B \subset A$$

Ex1. Submulțime maximală

Problemă: Se dă o mulțime de **n** numere întregi. Să se găsească o submulțime de dimensiune **k**, cu proprietatea că suma elementelor mulțimii este maximală.

Rezolvare:

v1

- Algoritmul execută k paşi
- La fiecare pas se alege maximul (optim local) din mulțime. Apoi se elimină din mulțimea inițială.

v2

- Algoritmul execută n-k paşi
- La fiecare pas se elimină din mulțime minimul. După n-k paşi mulțimea rămasă este submulțimea căutată

Ex2. Planificarea activităților

Problemă: Se dau **n** activități care partajează o resursă. Se doreşte alegerea unui număr maximal de activități care să nu se suprapună în partajare. O activitate este definită printr-un interval **[ti,tf]** în care are nevoie să acceseze resursa

Rezolvare:

- Se sortează activitățile după tf
- La fiecare pas se alege activitatea cu tf minim şi se elimină activitățile cu care se intersectează
- Algoritmul ciclează până nu mai sunt activități

Ex3. Problema rucsacului

Problemă: O persoană dispune de un rucsac care poate suporta o greutate maximă **G**. Având la dispoziție **n** obiecte de diferite greutăți **g(i)** şi un cost/câştig **c(i)** asociat fiecărui obiect, se doreşte umplerea rucacului cu câştig/cost maxim. Se consideră că **obiectele pot fi tăiate.**

Rezolvare:

- Se sortează obiectele descrescător după c(i)/g(i)
- La fiecare pas se alege obiectul care maximizează raportul c(i)/g(i)
- Algoritmul ciclează până când rucsacul este plin sau până când obiectul următor ce maximizează raportul c(i)/g(i) nu intră în întregime în rucsac, caz în care obiectul este tăiat a.î. să se umple rucsacul

Greedy euristic

1. Plata unei sume de bani folosind cât mai puține bancnote/monede dintr-o mulțime dată – există o infinitate de bancnote/monede disponibile, inclusiv de valoarea 1.

2. Săritura calului: se dă o tablă de şah de dimensiune nxn. Având în vedere că un cal (piesă de şah ☺) se află în punctul de coordonate (1,1), să se găsească un şir de mutări ale calului astfel încât să treacă exact o singură dată prin fiecare căsuță şi să nu existe căsuță neatinsă.

---

3. Codificare Huffman (studiu individual)