Sume podskupova i 0/1 ruksak

Ivan Laković

Problem sume podskupova

- Zadan nam je skup S od n brojeva, te gornja ograda W.
- Želimo izabrati podskup $P \subseteq S$ tako da suma elemenata u S bude što bliža W.

Problem sume podskupova

- Formalno:
- Imamo skup S {1,2,...,n} elemenata i svaki ima nenegativnu težinu w_i za svaki i=1,2,...n.
- Zadana nam je i granica $W \in \mathbb{R}$.
- Želimo pronaći $P \subseteq S$ takav da je $\sum_{i \in S} w_i \leq W$ i $\sum_{i \in S} w_i$ je najveća moguća.

Pohlepan pristup?

- Poredamo elemente silazno?
- {W/2+1, W/2, W/2}

- Poredamo elemente uzlazno?
- {1, W/2, W/2}

Problem 0/1 ruksaka

- Imamo ruksak koji može podnijeti težinu W i n predmeta, svaki vrijednosti v_i i težine w_i.
- Želimo napuniti ruksak tako da ima najveću moguću vrijednost.

Problem 0/1 ruksaka

- Formalno:
- Imamo skup S = {1,2,...,n} elemenata, te svaki ima nenegativnu težinu w_i, i vrijednost v_i za i=1,2,...n.
- Zadana nam je i granica $W \in \mathbb{N}$.
- Želimo pronaći $P \subseteq S$ takav da je $\sum_{i \in S} w_i \leq W$
- i $\sum_{i \in S} v_i$ je najveća moguća.

Rješenje (1):

- Prvi pristup, prođemo kroz sve kombinacije predmeta i pronađemo optimalnu.
- Vremenska složenost $O(2^n)$.

Rješenje:

- Neka je V[i, w] optimalno rješenje problema (ima najveću vrijednost) za svaki $i \leqslant n$, i $w \leqslant W$ s podskupom elemenata {1,2,...,i} takvo da mu je suma težina w_i manja od w.
- $V[i,j] = \max_P \sum_{j \in P} v_i$, gdje je $P \subseteq \{1,2,\ldots,i\}$ i zadovoljava $\sum_{j \in P} w_j \leq w$

Rješenje:

- Tada je V[i, w] za i=n i w=W traženo rješenje.
- Za V[n, W] vrijedi:
  - V[n, W] = V[n-1, W]; ako se n ne nalazi u optimalnom rješenju.
  - V[n, W] = V[n-1, W-w_n] + v_n; ako se n nalazi u optimalnom rješenju.

Rješenje:

- Dobivamo slijedeću relaciju:

$$V[i,w] = \begin{cases} 0 & ako\ je\ \ i=0\ \ ili\ \ w=0 \\ V[i-1,w] & ako\ je\ \ w<w_i \\ max\{V[i-1,w], V[i-1,w-w_i]+v_i\} & ako\ je\ \ i>0\ \ i\ \ w\geq w_i \end{cases}$$

Rješenje (2):

- Iz predhodne formule može se direktno napisati rekurzija.
- Vremenska složenost rekurzije $O(2^n)$.

Rekurzija:

```
int nadi_optimalan(int *vrijednosti, int *tezine, int broj_stvari, int max_tezina)
{
    if(broj_stvari == 0 || max_tezina <=0)
        return 0;

    if(tezine[broj_stvari-1] > max_tezina)
        return nadi_optimalan(vrijednosti, tezine, broj_stvari-1, max_tezina);
    else
    {
        int a, b;
        a = nadi_optimalan(vrijednosti, tezine, broj_stvari-1, max_tezina);
        b = nadi_optimalan(vrijednosti, tezine, broj_stvari-1,
           max_tezina-tezine[broj_stvari-1]) + vrijednosti[broj_stvari-1];
        return ( a > b ) ? a : b;
    }
}
```

Rekurzija:

Neka imamo težine (1, 1, 2), vrijednosti (5, 10, 20), te najveću moguću težinu W=3.

```
                          V(3, 3)       ---------> V(n, W)
                      /              \
                    /                  \
              V(2,3)                  V(2,1)
            /        \                /     \
          /            \            /         \
    V(1,3)        KV(1,2)     V(1,1)     V(1,0)
      / \            / \          / \
    /     \        /     \      /     \
V(0,3)  V(0,2)   V(0,2)  V(0,1)   V(0,1)   V(0,0)
```

Rješenje (3):

- Pristup preko dinamičkog programiranja.
- Za razliku od rekurzivnog samo se jednom računa svaki podproblem.
- Do krajnjeg rješenja dolazimo računanjem od dolje (bottom-up).
- Rješenje svakog podproblema je optimalno.

Rješenje (3):

- Primjer:
- Imamo ruksak veličine 9, te 4 predmeta veličine 2, 3, 4, 5 s vrijednostima 3, 4, 5, 7 respektivno.
- Treba pronaći optimalan vrijednost koju možemo ponijeti.

Težina ruksaka

Broj predmeta u ruksaku

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1										
2										
3										
4										

Težina ruksaka

Broj predmeta u ruksaku

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	3	3	3	3	3	3	3
2										
3										
4										

Težina ruksaka

Broj predmeta u ruksaku

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	3	3	3	3	3	3	3
2	0	0	3	4	4	7	7	7	7	7
3										
4										

Težina ruksaka

Broj predmeta u ruksaku	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	3	3	3	3	3	3	3
2	0	0	3	4	4	7	7	7	7	7
3	0	0	3	4	4	7	8	9	9	12
4										

Težina ruksaka

Broj predmeta u ruksaku

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	3	3	3	3	3	3	3
2	0	0	3	4	4	7	7	7	7	7
3	0	0	3	4	4	7	8	9	9	12
4	0	0	3	4	5	7	8	10	11	12

Rješenje (3):

- Na primjeru vidimo da je vremenska složenost O(n*W). To se naziva pesudo-polinomna složenost.
- Na primjeru vidimo da za prostornu složenost dovoljna O(W).

Ispunjavanje tablice:

```
void ispuni_tablicu(int **tablica, int *tezine, int *vrijednosti,
                                    int max_tezina, int broj_stvari)
{
    int i, j;
    for(i=0; i<max_tezina+1; ++i)
        tablica[0][i] = 0;

    for(i=1; i<broj_stvari+1; ++i)
        for(j=0; j<max_tezina+1; ++j)
        {
            tablica[i][j] = tablica[i-1][j];
            if(tezine[i-1] <= j  && tablica[i][j] < tablica[i-1][j-tezine[i-1]]
                                                            +vrijednosti[i-1])
                tablica[i][j] = tablica[i-1][j-tezine[i-1]]+vrijednosti[i-1];
        }
}
```

Ispunjavanje tablice s uštedom:

```
void ispuni_tablicu_stednja(int **tablica_stednja, int *tezine, int
                                        *vrijednosti, int max_tezina, int broj_stvari)
{
    int i, j;
    for(i=0; i<max_tezina+1; ++i)
        tablica_stednja[0][i] = 0;

    for(i=1; i<broj_stvari+1; ++i)
        for(j=0; j<max_tezina+1; ++j)
        {
            tablica_stednja[i%2][j] = tablica_stednja[(i-1)%2][j];
            if(tezine[i-1] <= j && tablica_stednja[i%2][j] <
                                tablica_stednja[(i-1)%2][j-tezine[i-1]]+vrijednosti[i-1])
                tablica_stednja[i%2][j] = tablica_stednja[(i-1)%2][j-tezine[i-1]]
                                                                        +vrijednosti[i-1];
        }
}
```

Literatura:

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein; "Introduction to Algorithms".

2. Jon Kleinberg, Éva Tardos; "Algorithm Design".

3. M. H. Alsuwaiyel; "Algorithms: Design Techniques and Analysis".

4. http://www.geeksforgeeks.org/dynamic-programming-set-10-0-1-knapsack-problem/

Pitanja?