ASD Laboratorio 08

Alessio Guerrieri/Lorenzo Ghiro

07/03/2018

Calendario

07/03	Dinamica 2
21/03	Progetto Dinamica
16/05	Approssimazione
23/05	Progetto Finale
30/05	Progetto Finale

Progetto Dinamica

- Programmazione dinamica;
- Assumiamo gli stessi gruppi del primo semestre, in caso di cambiamenti, avvisare **entro il 14/03**;

Sottosequenza crescente

Data una sequenza di interi scegliere un sottoinsieme della sequenza in modo che:

- gli elementi del sottoinsieme, messi nell'ordine in cui si trovavano nella sequenza originaria, formino una sequenza crescente
- il sottoinsieme abbia somma massima

Sottoproblema

S(i) = somma della sottosequenza crescente di somma massima a partire dall'elemento i

NON FUNZIONA! Per scegliere ottimamente, abbiamo bisogno di sapere l'ultimo elemento scelto

Sottosequenza crescente

Sottoproblema

$S[i, j]$= somma ottimale ottenibile dal sottoarray $[i..N-1]$ avendo scelto per ultimo l'elemento j

$$S[i,j] = \begin{cases} 0, & \text{if } i == n \\ S[i+1, j], & \text{if } A[i] < A[j] \\ max(S[i+1, j], S[i+1, i] + A[i]) & \text{if } A[i] \geq A[j] \end{cases}$$

Sottosequenza crescente

Sottoproblema alternativo

$S[i]$ =somma ottimale da i in poi essendo obbligati a scegliere l'elemento i

$$S[i] = A[i] + max_{j:(j>i,\ A[j] \geq A[i])}(S[j])$$

La soluzione del problema é uguale a $max(S)$

Zaino

Funzione di ricorrenza ($v[i]$: valore dell'i-esimo elemento, $p[i]$: peso dell'i-esimo elemento)
$S(C, i)$ = massimo valore ottenibile utilizzando gli elementi da i in poi, con uno zaino avente spazio C.

$$S(C,i) = \begin{cases} -\inf & \text{if } C < 0 \\ 0 & \text{if } i == N \\ Max \begin{cases} v[i] + S(C - p[i], i+1) \\ S(C, i+1) \end{cases} & \text{if } i < N \end{cases}$$

ZAINO RICORSIVO

```
1     int ric(int c,int i){
2       if(c<0)
3         return -100000000;
4       if(i==N)
5         return 0;
6       int p=elements[i].first;
7       int v=elements[i].second;
8       return max(v+ric(c-p,i+1),
9                  ric(c,i+1));
10    }
```

Nota: l'ordine dei casi base è importante

ZAINO MEMOIZATION

```
1   int ric(int c,int i){
2     if(c<0)
3       return -100000000;
4     if(i==N)
5       return 0;
6     if(sav[c][i]==-1){
7       int p=elements[i].first;
8       int v=elements[i].second;
9       sav[c][i]= max(v+ric(c-p,i+1),
10                     ric(c,i+1));
11    }
12    return sav[c][i];
13  }
```

ZAINO ITERATIVO

- Il calcolo di S(c,i) dipende dagli S(c',i+1).
- Calcoliamo prima tutti gli S(_,N-1), poi tutti gli S(_,N-2)...

```
1       for(int i=N-1;i>=0;i--){
2         int p=elements[i].first;
3         int v=elements[i].second;
4         for(int c=0;c<=C;c++){
5           if(elements[i].first<=c)
6             sav[c][i]=max(sav[c][i+1],
7                           v+sav[c-p][i+1]);
8           else
9             sav[c][i]=sav[c][i+1];
10          }
11        }
12      }
```

ZAINO ITERATIVO EFFICENTE

- Una volta calcolati tutti gli S(_,i), gli S(_,i+1) non ci servono piu.
- Utilizziamo un array C*2

```
1       for(int i=N-1;i>=0;i--){
2         int p=elements[i].first;
3         int v=elements[i].second;
4         int cur=i%2; int next=(i+1)%2;
5         for(int c=0;c<=C;c++){
6           if(elements[i].first<=c)
7             sav[c][cur]=max(sav[c][next],
8                             v+sav[c-p][next]);
9           else
10            sav[c][cur]=sav[c][next];
11        }
12      }
```

PILLOLE

$S[i,j]$ = numero di combinazioni ottenibili da una bottiglia contenente i pillole intere e j pillole smezzate

$$S[i,j] = \begin{cases} 1, & \text{if } i == 0 \text{ and } j == 0 \\ S[i-1, j+1], & \text{if } i > 0 \text{ and } j == 0 \\ S[i, j-1], & \text{if } i == 0 \text{ and } j > 0 \\ S[i, j-1] + S[i-1, j+1], & \text{if } i > 0 \text{ and } j > 0 \end{cases}$$

Problemi (I)

Sottosequenza comune massimale

Date due stringhe di caratteri alfanumerici, calcolare una sottosequenza comune massimale (secondo la definizione delle slides di Montresor). Stamparne la lunghezza

Lettura di una stringa (libreria string):

```
1   string s;
2   in>>s;
```

Ottenere dimensione stringa e valore per un singolo carattere

```
1   int dim=s.size();
2   char c=s[2];
```

Problemi (II)

Definizione: Node-Cover

Un insieme $S \subseteq V$ di nodi é un Node-Cover se ogni arco nel grafo/albero ha almeno uno dei due nodi in S

Min Cover su albero

Dato un albero, trovare la dimensione del Node-Cover di dimensione minima.

Min cover su albero pesato

Dato un albero con pesi su i nodi, trovare il Node-Cover di peso minimo e stamparne il peso.

Problemi (III)

Fiera

Secondo progetto 2011/2012