Esercitazione di programmazione dinamica

- Concentratevi su PD – evitate per oggi approcci diversi!
- Stabilite chiaramente cosa volete calcolare, in italiano:
  Sia $DP[i]$ il valore massimo che si ottiene con i primi i ...
- Se non riuscite a scriverlo in italiano, fermatevi! Non l'avete capito
- Ragionate sulla sottostruttura ottima: una soluzione ottima per un sottoproblema può essere utilizzata per trovare la soluzione ottima di un problema più grande?
- Scrivete un'equazione ricorsiva DP che calcoli il valore di una soluzione ottima a partire dal valore della soluzione più piccola
- Trasformate l'equazione ricorsiva in codice utilizzando programmazione dinamica o memoization

Ottimizza la somma

Supponete di avere in input un vettore V contenente n interi positivi distinti e un valore intero W. Scrivere un algoritmo

int bestSum(**int**[ ] V, **int** n, **int** W)

che

1. restituisca il massimo valore $X = \sum_{i=0}^{n-1} x[i]V[i]$ tale che $X \leq W$ e ogni $x[i]$ è un intero non negativo;
2. stampi il vettore x.

Ad esempio, per `V = [18, 3,21, 9,12,24]` e $W = 17$, una possibile soluzione ottima è `x = [ 0, 2, 0, 1, 0, 0]` da cui deriva $X = 2 \cdot 3 + 1 \cdot 9 = 15$.

Discutere correttezza e complessità.

Mosse su scacchiera

- Sia P una matrice di dimensione $n \times n$ che rappresenta una scacchiera in cui ogni casella (r, c) è associata a un profitto $P[r][c]$.
- Considerate un pedone che si può muovere (1) verso l'alto, oppure (2) in diagonale alto-destra, oppure (3) in diagonale alto-sinistra.
- Il pedone parte da una qualunque casella della riga inferiore e deve raggiungere una qualunque casella della riga superiore.
- Quando il pedone visita una casella (r, c), guadagna il profitto $P[r][c]$.

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

Mosse su scacchiera

Scrivere un algoritmo

int searchPath(**int**[][] P, **int** n)

che restituisca il massimo profitto ottenibile (calcolato come la somma dei profitti delle caselle visitate) facendo partire un pedone dalla riga inferiore e raggiungendo una qualunque casella della riga, rispettando le regole di movimento del pedone.

Alcuni suggerimenti:

- Nella matrice/scacchiera, dov'è l'alto? Dov'è il basso?
  Lo dovete decidere voi!
- La tabella DP è una matrice

I Promessi Sposi

"Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte; e il ponte, che ivi congiunge le due rive, par che renda ancor più sensibile all'occhio questa trasformazione, e segni il punto in cui il lago cessa, e l'Adda rincomincia, per ripigliar poi nome di lago dove le rive, allontanandosi di nuovo, lascian l'acqua distendersi e rallentarsi in nuovi golfi e in nuovi seni."

Quante volte questo testo contiene la sottosequenza `"lucia"`?

I Promessi Sposi

"Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte; e il ponte, che ivi congiunge le due rive, par che renda ancor più sensibile all'occhio questa trasformazione, e segni il punto in cui il lago cessa, e l'Adda rincomincia, per ripigliar poi nome di lago dove le rive, allontanandosi di nuovo, lascian l'acqua distendersi e rallentarsi in nuovi golfi e in nuovi seni."

Quante volte questo testo contiene la sottosequenza`"lucia"`?

Alcune considerazioni:

- Due sottosequenze sono distinte (e quindi vanno contate separatamente) se corrispondono a due sequenze diverse di indici del testo, anche se producono la stessa stringa di caratteri.
- Esempio: `"did you go"` contiene due volte la sottosequenza `"dog"`

I Promessi Sposi

Scrivere un algoritmo

int lucia(ITEM[] T, ITEM[] P, **int** n, **int** m)

che

- prenda in input
  - una stringa testo T lunga n caratteri;
  - una stringa pattern P lunga m caratteri;
- restituisca il numero di volte che la stringa P appare come sottosequenza distinta di T.

I Promessi Sposi

Scrivere un algoritmo

int lucia(ITEM[] T, ITEM[] P, **int** n, **int** m)

che

- prenda in input
  - una stringa testo T lunga n caratteri;
  - una stringa pattern P lunga m caratteri;
- restituisca il numero di volte che la stringa P appare come sottosequenza distinta di T.

Suggerimenti:

- Fate attenzione al ruolo non-simmetrico di i e j: cercare `"dog"` in una stringa vuota è diverso da cercare una stringa vuota in `"dog"`