Progettazione di Algoritmi **Anno Accademico 2021–2022**

Esercizi sulla Programmazione Dinamica.

Ugo Vaccaro

1. *Esercizio:* Dati interi k, n, con $0 \leq k \leq n$, i numeri di Stirling del secondo tipo sono denotati con il simbolo $\left\{ {n \atop k} \right\}$, definiti dalla equazione di ricorrenza

$$\left\{ {n \atop k} \right\} = k \left\{ {n-1 \atop k} \right\} + \left\{ {n-1 \atop k-1} \right\}$$

e dalle condizioni iniziali

$$\left\{ {0 \atop 0} \right\} = 1, \qquad \left\{ {n \atop 0} \right\} = 0 \quad \text{per } n > 1, \quad \text{e} \qquad \left\{ {n \atop n} \right\} = 1, \quad \forall n.$$

Si scriva un algoritmo di Programazione Dinamica che, prendendo in input interi k, n, con $0 \leq k \leq n$, restituisca in output il corrispondente numero di Stirling $\left\{ {n \atop k} \right\}$. Si analizzi la complessità dell'algoritmo.

2. *Esercizio:* Dati interi k, n, con $0 \leq k \leq n$, i numeri di Eulero sono denotati con il simbolo $\left\langle {n \atop k} \right\rangle$, definiti dalla equazione di ricorrenza

$$\left\langle {n \atop k} \right\rangle = (k+1) \left\langle {n-1 \atop k} \right\rangle + (n-k) \left\langle {n-1 \atop k-1} \right\rangle, \qquad n > 0$$

e dalle condizioni iniziali

$$\left\langle {n \atop 0} \right\rangle = 1, \quad \forall n \geq 0, \qquad \left\langle {n \atop n} \right\rangle = 0, \quad \forall n > 0.$$

Si scriva un algoritmo di Programazione Dinamica che, prendendo in input interi k, n, con $0 \leq k \leq n$, restituisca in output il corrispondente numero di Eulero $\left\langle {n \atop k} \right\rangle$. Si analizzi la complessità dell'algoritmo.

3. *Esercizio:* Dati interi $i \geq 0, j \geq 0$, i numeri di Shor sono definiti dall'equazione di ricorrenza

$$S(i,j) = iS(i-1,j) + (i+j)S(i-1,j-1), \text{ e } \ S(0,j) = 1 = S(i,0), \ \forall i,j.$$

Si scriva un algoritmo di Programazione Dinamica che, prendendo in input interi i, j, con $i \geq 0, j \geq 0$, restituisca in output il corrispondente numero di Shor $S(i,j)$. Si analizzi la complessitá dell'algoritmo.

4. *Esercizio:* Per ogni intero $n \geq 0$, i numeri di Catalano C_n sono definiti dall'equazione di ricorrenza

$$C_n = \sum_{i=0}^{n-1} C_i \times C_{n-i-1},$$

con la condizione iniziale che $C_0 = 1$. Si scriva un algoritmo di Programazione Dinamica che, prendendo in input l'intero n, con $0 \leq n$, restituisca in output il corrispondente numero di Catalano C_n. Si analizzi la complessità dell'algoritmo.

5. *Esercizio:* Dato un intero n, progettare ed analizzare un algoritmo di programmazione dinamica che conti il numero di modi di esprimere n come somma degli interi $1, 3, 4$. Ad esempio, se

$n = 5$, la risposta è 6, in quanto

$$\begin{aligned} 5 &= 1+1+1+1+1 \\ &= 1+1+3 \\ &= 1+3+1 \\ &= 3+1+1 \\ &= 1+4 \\ &= 4+1 \end{aligned}$$

6. *Esercizio:* Si supponga di dover stendere un cavo di rete lungo metri n. Il cavo si puó comprare in pezzi da metri k, per ciascun $k \in \{1, \ldots, n\}$. Comprare un pezzo di cavo lungo k costa c_k. Per motivi a noi ignoti NON possiamo assumere che $c_{k+1} \geq c_k$. Progettare ed analizzare un algoritmo di programmazione dinamica che prendendo in input i valori $c_1, \ldots, c_n$ determini il minimo costo per realizzare il cavo lungo n.

7. *Esercizio:* Si progetti e si analizzi un algoritmo di Programmazione Dinamica per il seguente problema:
Input: una sequenza di caratteri `a=a[1]...a[n]`,
Output: La lunghezza della più lunga sottosequenza di `a=a[1]...a[n]` (composta da caratteri *non* necessariamente consecutivi in `a=a[1]...a[n]`) che è palindroma.

   Esempio: se `a=ABBDCACB`, la sua più lunga sottosequenza palindroma sarebbe `BCACB`, e quindi l'algoritmo dovrebbe restituire il valore 5.

8. *Esercizio:* Si progetti e si analizzi un algoritmo di Programmazione Dinamica per il seguente problema:
Input: Una barra lunga `n` metri, un vettore di profitti `p=p[1]...p[n]`, dove il generico valore `p[i]` ci dice quanto guadagniamo dal tagliare dalla barra un pezzo di `i` metri.
Output: Massimo profitto che si può ottenere dalla barra, tagliando da essa pezzi opportuni.

   Esempio: se `p=p[1]p[2]p[3]p[4]=1 5 8 9`, potremmo tagliare la barra in un solo pezzo di 4 metri, di profitto 9, oppure tagliare la barra in un pezzo da 1 metro ed uno da 3 metri, con profitto totale 1+8=9, oppure tagliare la barra in un pezzo da 2 metri ed un altro da 2 metri, con profitto totale 5+5=10, oppure in un pezzo da 1 metro, un altro di 1 metro ed uno di 2 metri, con profitto totale 1+1+5=7, oppure tagliare dall abarra 4 pezzi di 1 metro, con profitto totale 1+1+1+1=4, etc.

9. *Esercizio:* Sia $E = \{e_1, e_2, \ldots, e_n\}$ l' insieme degli n esami attivati nel Corso di Laurea in Informatica. Ogni esame e_i vale c_i crediti formativi ed ha un grado di difficoltà rappresentato da un coefficiente d_i. Gli studenti possono creare il loro piano di studi individuale scegliendo nella lista degli esami attivati un sottoinsieme di esami tale che la somma dei crediti corrispondenti sia almeno un dato valore C.

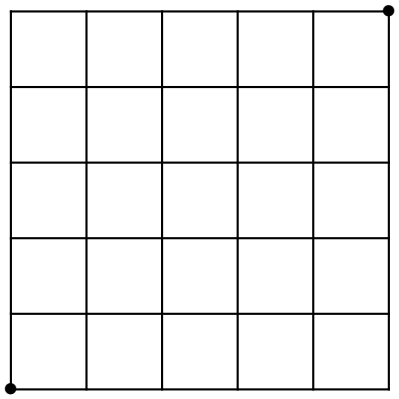
   Progettare ed analizzare un algoritmo di Programmzione Dinamica che determini la minima difficoltà totale di piani di studi aventi somma di crediti totale almeno C.

10. *Esercizio:* È dato un esame con n esercizi $e_1, \ldots, e_n$. Per ogni $i = 1, \ldots, n$ l'esercizio i-esimo vale p_i punti e richiede m_i minuti per essere risolto. Dato in input il vettore contenente i valori p_i, $i = 1, \ldots, n$ ed il vettore dei minuti m_i, $i = 1, \ldots, n$, si calcoli il *mininimo* numero di minuti necessari per risolvere esercizi di punteggio totale P.

    [Suggerimento: Sia $M(i, p)$ il minimo numero di minuti richiesto per ottenere p punti, risolvendo esercizi tra i primi i: $e_1, \ldots, e_i$ (noi siamo interessati a $M(n, P)$). Per fare ciò, si può risolvere l'esercizio i-esimo, o non risolverlo. Il resto è facile....]

11. *Esercizio:* Descrivere il problema del calcolo della Distanza di Edit tra due sequenze studiato a lezione. Derivare, ed analizzare, per esso un algoritmo di Programmazione Dinamica. Si argomentino le affermazioni fatte.

12. *Esercizio:* Si supponga di avere una griglia di dimensioni $n \times n$. Si parte dal punto disposto nell'angolo in basso a sinistra della griglia. L'arrivo è il punto in alto a destra della griglia. Ad esempio, di sotto è rappresentata una griglia di dimensione 6×6. I punti di inizio ed arrivo sono evidenziati con un $\bullet$.



Si parte dal punto disposto nell'angolo in basso a sinistra della griglia. L'arrivo è il punto in alto a destra della griglia. Ad ogni passo, o si sale nel punto immediatamente di sopra nella stessa colonna, oppure ci si sposta nel punto immediatamente a destra nella stessa riga. Quanti possibili modi ci sono per andare dal punto di partenza all'arrivo? Progettare ed analizzare un algoritmo di Programmazione Dinamica che effettui il calcolo sopra richiesto.

[Suggerimento: si decomponga il calcolo richiesto come la somma dei possibili cammini a seconda che il primo passo sia verso l'alto o verso destra].

13. *Esercizio:* Siano dati un intero positivo T ed un insieme di interi positivi $\mathcal{A} = \{a_1, \ldots, a_k\}$. Progettare un algoritmo di Programmazione Dinamica che ritorni il valore TRUE se esiste o meno un sottoinsieme $\mathcal{B} \subseteq \mathcal{A}$ tale che $T = \prod_{b \in \mathcal{B}} b$, il valore FALSE nel caso contrario.

14. *Esercizio:* Siano dati un intero positivo T ed un insieme di interi positivi $\mathcal{A} = \{a_1, \ldots, a_k\}$. Progettare un algoritmo di Programmazione Dinamica che conti il numero di sottoinsiemi $\mathcal{B} \subseteq \mathcal{A}$ tale che $T = \sum_{b \in \mathcal{B}} b$.

[Suggerimento: si proceda come nel calcolo delle combinazioni $\binom{n}{k}$, che abbiamo espresso come $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$. Qui ovviamente è diverso, voglio solo dire che occorre calcolare il numero richiesto a seconda se un dato numero in $\mathcal{A}$ lo usiamo o non lo usiamo.]

15. *Esercizio:* Si consideri una scacchiera quadrata rappresentata da una matrice `M[1..n,1..n]`. Scopo del gioco è spostare una pedina dalla casella di partenza in alto a sinistra di coordinate `(1,1)` alla casella di arrivo in basso a destra di coordinate `(n, n)`. Ad ogni mossa la pedina può essere spostata di una posizione verso il basso oppure verso destra (senza uscire dai bordi della scacchiera). Quindi, se la pedina si trova in `(i,j)` potrà essere spostata in `(i+1,j)` oppure `(i,j+1)`, *se possibile*. Ogni casella `M[i,j]` contiene un numero reale; man mano che la pedina si muove, il giocatore accumula il punteggio segnato sulle caselle attraversate, incluse quelle di partenza e di arrivo. Progettare ed analizzare un algoritmo di Programmazione Dinamica che, data in input la matrice `M[1..n,1..n]` restituisce il massimo punteggio che è possibile ottenere spostando la pedina dalla posizione iniziale a quella finale con le regole di cui sopra.

(Suggerimento. Si usi una matrice `P[1..n,1..n]` di numeri reali, tale che in `P[i,j]` ci sia il massimo punteggio che è possibile ottenere spostando la pedina dalla cella `(1,1)` fino alla cella `(i,j)`. Una volta calcolati tutti i valori degli elementi di `P`, la risposta al nostro problema sarà

il valore contenuto in `P[n,n]`. Si esprima `P[i,j]` in funzione dei valori di `P` relativi alle celle da cui si può raggiungere la cella `(i,j)` con una mossa ed in funzione della matrice `M`.)

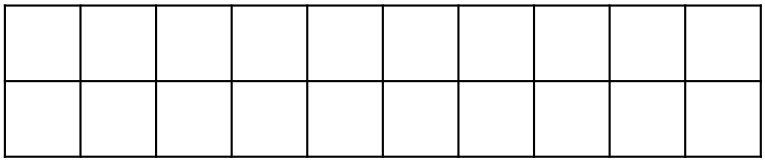
16. *Esercizio:* Si consideri la seguente variante del problema dello Zaino 0/1. L'input é costituito da n oggetti $a_1 \ldots, a_n$ di peso $w_1, w_2, \ldots, w_n$, e di valore $v_1, v_2, \ldots, v_n$, rispettivamente. Per ogni possibile $v = 1, \ldots, nV$, dove $V = \max\{v_1, \ldots, v_n\}$, e possibile indice $i = 1, \ldots, n$, si intende calcolare la quantità

    $w(i, v) =$ minimo peso di un sottoinsieme di $\{a_1, \ldots, a_i\}$ di profitto *esattamente uguale* a v

    (si puó assumere, per convenzione, che se non esiste un sottoinsieme di $\{a_1, \ldots, a_i\}$ di profitto uguale a v, allora $w(i, v) = \infty$).

    Si determini una equazione di ricorrenza per i valori $w(i, v)$, e si presenti un algoritmo di Programmazione Dinamica per il loro calcolo.

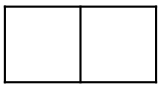
17. *Esercizio:* Si supponga di avere una stanza le cui dimensioni sono 2 di due metri per n metri. Ad esempio, di sotto avremmo la rappresentazione di una stanza di di 2 metri per 10 metri.

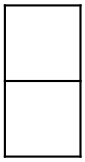
    Abbiamo altresì a disposizione piastrelle di dimensione 1 metro per 2 metri, del tipo

    Ciascuna piastrella può essere disposta sia orizzontalmente, ovvero così:

    oppure vertcalmente, ovvero così:

    Si conti il numero di modi con cui si può piastrellare la stanza in questione.

    [Suggerimento: se si inizia con la piastrella disposta verticlalmente, che problema occorre poi risolvere? Se si inizia con la piastrella disposta orizzontalmente, bisognerà necessariamente metterne un'altra orizzontalmente. Perchè? Che sottoproblema poi rimane?]

18. *Esercizio:* Si consideri la seguente variante del problema dello Zaino 0/1. L'input è costituito da oggetti di peso $w_1, w_2, \ldots, w_n$, di valore $v_1, v_2, \ldots, v_n$, e da una capacità di carico totale W. Si chiede di determinare il massimo valore di una collezione di oggetti, di peso totale al più W, sotto la condizione che di ciascun oggetto ne possiamo prendere anche più di una copia (nella variante studiata a lezione di ciascun oggetto si poteva prendere al più una copia). Si determini una equazione di ricorrenza per la soluzione del problema in questione, e si presenti un algoritmo di Programmazione Dinamica per il loro calcolo.

19. *Esercizio:* Descrivere il problema del calcolo del sottoinsieme di attività mutualmente compatibili di valore massimo studiato a lezione. Derivare, ed analizzare, per esso un algoritmo di Programmazione Dinamica. Si argomentino le affermazioni fatte.

20. Esercizio*:* Descrivere il problema del Cambio delle Monete studiato a lezione. Derivare, ed analizzare, per esso un algoritmo di Programmazione Dinamica. Si argomentino le affermazioni fatte.

21. Esercizio: Descrivere il problema del calcolo della più lunga sottosequenza comune a due sequenze studiato a lezione. Derivare, ed analizzare, per esso un algoritmo di Programmazione Dinamica. Si argomentino le affermazioni fatte.

22. *Esercizio*: Descrivere il problema del calcolo della distanza di edit studiato a lezione. Derivare, ed analizzare, per esso un algoritmo di Programmazione Dinamica. Si argomentino le affermazioni fatte.

23. *Esercizio*: Date due sequenze $s = s[1] \dots s[n]$ e $p = p[1] \dots p[m]$, diremo che p è una sottosequenza di s se esistono indici $1 \leq i_1 < i_2 < \ldots < i_m \leq n$ tali che $s[i_1] = p[1], s[i_2] = p[2], \ldots, s[i_m] = p[m]$.

    Il problema che vogliamo studiare è quello di calcolare il numero di volte che p compare come sottosequenza di s.

    Derivare, ed analizzare, per esso un algoritmo di Programmazione Dinamica. Si argomentino le affermazioni fatte.

24. *Esercizio*: Consideriamo un gioco in cui vi sono n monete di valori $v_1, \ldots, v_n$, rispettivamente, disposte su di una riga da sinistra a destra (nell'ordine indicato), con n pari, e due giocatori: Alice e Bob. All'istante 1 Alice sceglie una delle due monete all'estremità della riga (quindi, all'inizio del gioco Alice può scegliere una tra le monete di valore v_1 e v_n), e la rimuove dalla riga. All'istante 2 Bob sceglierà una delle due monete all'estremità della riga *rimanente* (quindi, Bob può scegliere una tra le monete di valore v_2 e v_n, se Alice ha scelto nel turno precedente la moneta di valore v_1, oppure Bob potrà scegliere una tra le monete di valore v_1 e v_{n-1}, se Alice ha scelto nel suo turno la moneta di valore v_n), e la rimuove dalla riga. Il gioco procede in questo modo, a turni alterni, fino a quando le monete sono esaurite. Vince il giocatore che, alla fine del gioco, ha scelto le $n/2$ monete di valore totale *massimo*.

    Il problema algoritmico che vogliamo risolvere è quello di determinare il valore massimo che il primo giocatore può ottenere (sotto l'ipotesi che anche il suo opponente giochi in maniera razionale).

    Progettare e analizzare un algoritmo di Programmazione Dinamica per il problema in questione. Si argomentino le affermazioni fatte.

25. *Esercizio.* Date due sequenze $a = a[1]...a[m]$ e $b = b[1]...b[n]$ di caratteri, il problema è di trovare la lunghezza della più corta supersequenza che contiene a e b come sottosequenze. Detto in altri termini, cerchiamo una sequenza $c = c[1]...c[k]$, con k *minimo*, tale che, per opportuni interi $1 \leq i_1 < i_2 < \ldots i_m \leq k$ e $1 \leq j_1 < j_2 < \ldots j_n \leq k$ valga che

$$c[i_1] = a[1], c[i_2] = a[2], \ldots, c[i_m] = a[m] \quad \text{e} \quad c[j_1] = b[1], c[j_2] = b[2], \ldots, c[j_n] = b[n]$$

    Derivare, ed analizzare, per esso un algoritmo di Programmazione Dinamica. Si argomentino le affermazioni fatte.