ESERCIZI SULLA TECNICA *Programmazione Dinamica*

1. **[SCACCHIERA]** Una pedina è posizionata sulla casella $(1, 1)$ in alto a sinistra di una scacchiera $n \times n$ e deve raggiungere la casella (n, n) in basso a destra. Quando posizionata sulla generica casella (i, j) per la pedina sono possibili al più due mosse: spostarsi verso il basso nella casella $(i + 1, j)$, posto che $i < n$, o spostarsi verso destra nella casella $(i, j + 1)$, posto che $j < n$. Stando così le cose la pedina raggiungerà la casella (n, n) in basso a destra con un cammino che l'ha portata a toccare $2n - 2$ caselle. Ad ogni casella della scacchiera è associato un valore $c_{i,j}$ ed il valore del cammino è dato dalla somma dei valori delle caselle toccate dalle pedina.
   Progettare un algoritmo che trova il cammino di valore massimo in $\mathrm{O}(n^2)$.

2. **[k-SOTTOSEQUENZA]** Si consideri una sequenza di interi X. La cancellazione da X di un certo numero di elementi determina una *sottosequenza*. Una k-sottosequenza di X è una sottosequenza di X in cui compaiono al più k elementi consecutivi di X.
   Data una sequenza X di n interi ed un intero k, con $k \leq n$ vogliamo trovare una k-sottosequenza di X la somma dei cui elementi sia massima.
   Ad esempio, per $X = (9, 8, 1, 9, 2, 9, 8, 2, 1, 8, 8, 1)$ e $k = 3$ la 3-sottosequenza a somma massima è $(9, 8, 9, 9, 8, 8, 8, 1)$ e vale 60.

   a. Descrivere un algoritmo che risolve il problema nel caso particolare $k = 2$ in $\mathrm{O}(n)$.

   b. Descrivere un algoritmo che risolve il problema nel caso generale in $\mathrm{O}(k \cdot n)$.

3. **[M-LISTE]** Data una matrice $n \times m$ di interi $M = [m_{i,j}]$. Una M-lista è una sequenza $(m_{1,j_1}, m_{2,j_2} \ldots m_{n,j_m})$ tale che $1 \leq j_1 \leq \ldots j_m \leq m$. Il valore di una M-lista è la somma degli elementi che la compongono.
   Progettare un algoritmo che trova un M-lista di valore minimo in $\mathrm{O}(n \cdot m^2)$.

4. **[SOTTOSEQUENZE CRESCENTI]** Si consideri una sequenza di interi X. La cancellazione da X di un certo numero di elementi determina una *sottosequenza*. La sottosequenza è crescente se il valore dei suoi elementi è crescente. La lunghezza della sottosequenza è il numero di elementi che la compongono. Il valore della sottosequenza è dato dalla somma dei valori degli elementi che la compongono.
   Data la sequenza $X = (x_1, x_2, \ldots x_n)$ a valori distinti si considerino i seguenti tre problemi:

   a. Trovare la sottosequenza crescente di X di lunghezza massima.
   Ad esempio per $X = (50, 4, 1002, 48, 3, 34, 30)$ la sottosequenza crescente di lunghezza massima è $(2, 3, 34)$.

   b. Trovare la sottosequenza crescente di X di valore massimo.
   Ad esempio per $X = (50, 2, 100, 1, 20, 30)$ la sottosequenza crescente di valore massimo è $(50, 100)$.

c. Trovare il numero delle sottosequenze crescenti di X.
Ad esempio per $X = (5, 3, 7, 8, 6)$ il numero di sottosequenze crescenti è 14 (le sottosequenze sono: (5), $(5, 7)$, $(5, 7, 8)$, $(5, 8)$, $(5, 6)$, (3), $(3, 7)$, $(3, 7, 8)$, $(3, 8)$, $(3, 6)$, (7), $(7, 8)$, (8), (6).

Progettare tre algoritmi che risolvono i tre problemi. La complessità degli algoritmi deve essere $O(n^2)$.

5. [**MASSIMO PRODOTTO DI SOTTOVETTORI**] Dato un vettore V di n numeri razionali si vuole un sottovettore (una sequenza di elementi consecutivi del vettore) il prodotto dei cui elementi sia massimo.

   a. Provare che, grazie ad un pre-processing, con il metodo della ricerca esaustiva è possibile risolvere il problema in $O(n^2)$.

   b. Progettare un algoritmo che risolve il problema in $O(n)$.

6. [**COMPRAVENDITA**] Dato un vettore P di n interi in cui $P[i]$ rappresenta il prezzo di una certa merce fra i giorni, ci piacerebbe sapere qual'è il giorno i in cui conviene comprare quella merce ed il giorno j, con $j > i$, in cui conviene rivenderla in modo da massimizzare il profitto o in alternativa (se non è possibile guadagnarci) minimizzare la perdita. In altre parole siamo interessati a conoscere la coppia (i, j) con $i < j$ per cui risulta massimo il valore $P[j] - P[i]$. Descrivere un algoritmo che risolve il problema in $O(n)$.

7. [**k-SOTTOVETTORI**] Dato un vettore V di n interi ed un intero k, $k \leq n$, si vuole un sottovettore (una sequenza di elementi consecutivi del vettore) di lunghezza massima contenente al più k elementi distinti.

   a. Descrivere un algoritmo che risolve il problema nel caso particolare $k = 2$ in $O(n)$.

   b. Descrivere un algoritmo che risolve il problema nel caso generale in $O(k \cdot n)$.

8. [**PALINDROMI** Una stringa è palindroma se non cambia leggendola da sinistra a destra o viceversa.
   Data una stringa $S = a_1 a_2 \dots a_n$ di n caratteri si considerino i seguenti tre problemi:

   a. Calcolare il minimo numero di caratteri che occorre inserire per rendere S palindroma.

   b. Determinare la lunghezza della più lunga sottostringa palindroma di S.

   c. Determinare il numero minimo m di sottostringhe palindromi $T_1, T_2, \dots, T_m$ tali che $S = T_1 \cdot T_2 \dots T_m$.

   Progettare tre algoritmi che risolvono i tre problemi. La complessità degli algoritmi deve essere $O(n^2)$.

9. [**ESAMI**] Uno studente in questo semestre può seguire n corsi e conta di dedicare allo studio di questi corsi un numero di ore pari a k. È importante per lui superarne almeno uno. Il problema sta quindi nel ripartire le k ore di studio tra i diversi corsi in modo da minimizzare la probabilità di non superarne alla fine neppure uno. Per decidere quanto studiare per ciscun corso lo studente dispone di una matrice P dove $P[i, j]$, con $1 \leq i \leq n$ e $0 \leq j \leq k$, rappresenta la probabilità di fallire il corso i avendogli dedicato j ore di studio. Descrivere un algoritmo che indica allo

studente quante delle k ore dedicare a ciascuna materia in modo da minimizzare la probabilità di non superarne alcuna (si ricorda che se p_i è la probabilità di fallire nel corso i allora la probabilità di fallire in tutti i corsi è $\prod_{1 \leq i \leq n} p_i$). La complessità dell'algoritmo deve essere $O(n \cdot k^2)$.

Ad esempio per $n = 3$, $k = 4$ e matrice P delle probabilità:

$$\begin{pmatrix} .8 & .7 & .65 & .62 & .6 \\ .75 & .7 & .67 & .65 & .62 \\ .9 & .7 & .6 & .55 & .5 \end{pmatrix}$$

La strategia migliore è spendere un ora sul primo corso e tre sul terzo (trascurando quindi il secondo). In questo modo la probabilità di non superare nessuno dei tre corsi è di circa il 29%.

10. [**PARENTESIZZAZIONE 1**] Si consideri il seguente problema. Data una espressione

$$E = c_1 O_1 c_2 O_2 \dots c_{n-1} O_{n-1} c_n$$

dove c_i è un intero positivo, per $1 \leq i \leq n$ e $O_j \in \{+, \cdot\}$ per $1 \leq j < n$. Inserire una serie di parentesi in modo che il valore dell'espressione calcolato rispettando la parentesizzazione introdotta risulti minimo. Descrivere un algoritmo che, data l'esepressione E, determina il valore minimo ottenibile tramite una sua parentesizzazione. L'algoritmo deve avere complessità $\mathrm{O}(n^3)$.

11. [**PARENTESIZZAZIONE 2**] Si consideri il seguente problema. Data una espressione

$$E = c_1 / c_2 / \dots c_{n-1} / c_n$$

dove c_i è un numero razionale positivo, per $1 \leq i \leq n$ e / denota l'operazione di divisione. Inserire una serie di parentesi in modo che il valore dell'espressione calcolato rispettando la parentesizzazione introdotta risulti minimo. Descrivere un algoritmo che, data l'esepressione E, determina il valore minimo ottenibile tramite una sua parentesizzazione. L'algoritmo deve avere complessità $\mathrm{O}(n^3)$.

12. [**ALBERI DI RICERCA OTTIMALI**] Bisogna creare un albero di ricerca contenente n chiavi $k_1 < k_2 < \dots < k_n$, a ciascuna delle chiavi è associato un peso p_i rappresentante la frequenza con cui si stima che la chiave verrà poi ricercata. Assumendo che la radice dell'albero è a livello 1, il costo dell'albero di ricerca per le n chiavi viene definito come $\sum_{i=1}^{n} l_i \cdot p_i$ dove l_i è il livello dell'albero in cui si trova la chiave k_i.
Descrivere un algoritmo che in tempo $O(n^3)$ trova l'albero di ricerca di costo minimo.

13. [**CAMMINI PER DAG-UNIDIREZIONALI**] Un grafo diretto $G = (\{1, 2, \dots, n\}, E)$ con $E \subseteq \{(i, j) \mid 1 \leq i < j \leq n\}$ è detto *dag-unidirezionale*. Si considerino i seguenti problemi:

    a. Dato un dag-unidirezionale pesato e con pesi non necessariamente positivi, bisogna trovare, se esite, il cammino di costo minimo che dal vertice 1 porta al vertice n. Descrivere un algoritmo che risolve il problema in $\mathrm{O}(n^2)$.

    b. Dato un dag-unidirezionale $G = (\{1, 2, \dots n\}, E_0 \cup E_1)$ pesato e con pesi non necessariamente positivi, trovare un cammino alternante che dal vertice 1 porta al vertice n. Un cammino $(v_{i_1}, v_{i_2} \dots v_{i_k})$ è detto *alternante* se alterna archi di E_0 ad archi di E_1, ovvero se $(v_{i_j}, v_{i_{j+1}}) \in E_0$ per ogni j dispari mentre $(v_{i_j}, v_{i_{j+1}}) \in E_1$ per ogni j pari.

Progettare due algoritmi che risolvono i due problemi. La complessità degli algoritmi deve essere $O(n^2)$.

14. **[CAMMINI MINIMI]** Sia $G = (\{1, 2, \ldots, n\}, E)$ un grafo diretto e pesato con pesi positivi. II peso dell'arco $(i.j)$ rappresenta il costo per andare direttamente da i a j. Siamo interessati a conoscere per ogni coppia ordinata di vertici (i, j) il costo minimo per andare da i a j (vale a dire il minimo tra i costi dei cammini che da i portano a j). Qualora non esista un cammino che da i porta a j il costo della coppia ordinata (i, j) è $+\infty$. Descrivere un algoritmo che risolve il problema in $O(n^3)$.

15. **[SELEZIONE DI ATTIVITÀ PESATE]** Si hanno n attività e per ogni attività, $1 \leq i \leq n$, l'intervallo temporale $[s_i, f_i)$ in cui l'attività dovrebbe svolgersi e il guadagno v_i che si ottiene dallo svolgimento dell'attività. Due attività i e j sono compatibili se gli intervalli temporali $[s_i, f_i)$ e$[s_j, f_j)$ sono disgiunti ed il valore di un sottoinsieme di attività è dato dalla somma dei valori delle attività del sottoinsieme. Vogliamo selezionare un sottoinsieme S di valore massimo di attività tra loro compatibili. Descrivere un algoritmo che risolve il problema in $O(n^2)$.

16. **[ATTIVITÀ]** Date n attività (lezioni, seminari, ecc.) ognuna caratterizzata da un tempo di inizio e un tempo di fine si vuole trovare un sottoinsieme delle attività che possono essere svolte in un unica aula senza sovrapposizioni e che massimizzi il tempo totale di utilizzo dell'aula. Progettare un algoritmo che risolve il problema e valutarne la complessità.

17. **[MONETE]** Si dispone di un numero illimitato di monete di n diversi valori $1 = v_1, v_2, \ldots, v_n$ e bisogna dare un resto C utilizzando il minor numero di monete. Descrivere un algoritmo che risolve il problema in $O(n \cdot C)$.

18. **[ASSEGNAMENTO DI LAVORI]** Bisogna eseguire n lavori, ciascuno caratterizzato da un guadagno $\{v_1, v_2, \ldots, v_n\}$, un tempo richiesto per la sua esecuzione $\{t_1, t_2, \ldots, t_n\}$ ed una scadenza per la sua esecuzione $\{d_1, d_2, \ldots, d_n\}$, con t_i e d_i interi e $d_1 \leq d_2, \ldots \leq d_n$. Sapendo che il guadagno si ha solo se il lavoro viene svolto entro la sua scadenza e che si dispone di un'unica macchina, descrivere un algoritmo che calcoli il guadagno massimo che è possibile ottenere e l'ordine in cui gli n lavori vanno eseguiti per ottenerlo. La complessità dell'algoritmo deve essere $O(n \cdot d_n)$.

19. **[SOMMA DI SOTTOINSIEMI]** Progettare un algoritmo che, preso un insieme di interi positivi $A = \{a_1, a_2, \ldots, a_n\}$ ed un intero k trovi (se esiste) un sottoinsieme di A la somma dei cui elementi dia k. La complessità dell'algoritmo deve essere $O(k \cdot n)$.

20. **[PARTITA DI CALCETTO]** Si supponga di dover organizzare una partita di calcetto e di dover dividere i partecipanti in due squadre il più possibile bilanciate. In particolare sia $A = \{a_1, a_2, \ldots a_{2n}\}$ l'insieme dei partecipanti e per ogni $i = 1, 2, \ldots 2n$, sia v_i un coefficiente che identifica la bravura del giocatore a_i. Progettare un algoritmo per partizionare i $2n$ giocatori in due squadre da n in modo che la differenza tra la bravura delle due squadre sia minima. L'algoritmo deve avere complessità $O(n^2 \cdot M)$, dove M è il valore somma delle bravure degli n partecipanti.

21. **[DIVISIONE DI LAVORI]** Bisogna eseguire n differenti lavori e si dispone di due operai. Per le diverse attitudini dei due operai può accadere che un certo lavoro richieda un tempo diverso

a seconda dell'operaio cui viene assegnato. Sia $v_{i,j}$ con $1 \leq i \leq n$ e $j \in \{1,2\}$ un intero positivo rappresentante il tempo richiesto dal lavoro i se eseguito dall'operaio j. Descrivere un algoritmo che, dati i $2{\cdot}n$ valori $v_{1,1}, v_{1,2}, \ldots v_{n,1}, v_{n,2}$, assegni gli n lavori ai due operai in modo da minimizzare il tempo necessario alla loro esecuzione. La complessità dell'algoritmo deve essere $O(M_1 \cdot M_2 \cdot n)$. Dove M_i è il tempo richiesto dall'operaio i per eseguire tutti i lavori, $i \in \{1,2\}$.

22. **[FURTO]** Una banda di tre ladri deve spartirsi il frutto di una rapina di n oggetti $\{a_1, a_2, \ldots, a_n\}$, ciascuno caratterizzato da un proprio valore intero positivo v_i. Sapendo che l'ammontare totale del bottino M è divisibile per tre, descrivere un algoritmo che verifichi se è possibile spartire gli n oggetti in parti di ugual valore e, in caso affermativo, produca la spartizione. L'algoritmo deve avere complessità $O(n \cdot M^2)$.

23. **[TASSA]** Sia $c_1, c_2, \ldots, c_n$ una sequenza di città che devono essere visitate tutte e nell'ordine della sequenza. In ogni città c_i si deve pagare una tassa t_i. Inoltre in accordo al bit e_i si riceve o meno un bonus per l'esenzione dal pagamento di tasse successive. Ogni esenzione può essere usata una sola volta. Descrivere un algoritmo che, dati t_i ed e_i per ogni città c_i, determina la tassa totale minima che un viaggiatore deve pagare per attraversare tutte le n città. La complessità dell'algoritmo deve essere $O(n^2)$.

24. **[PIANI DI STUDIO]** Sia $A = \{a_1, a_2, \ldots a_n\}$ un insieme di n esami, dove per ogni $i = 1, 2, \ldots n$, l'esame a_i vale c_i crediti e si supponga di avere per ogni esame a_i un coefficiente d_i che rappresenta il grado di difficoltà dell'esame. Ogni studente può redigere il proprio piano di studio individuale scegliendo nella lista degli esami attivati un insieme di esami tali che la somma dei crediti corrispondenti sia almeno P. Progettare un algoritmo che redige un piano di studi regolare di difficoltà minima in $\mathrm{O}(n \cdot P)$.

25. **[TAGLIO IN SOTTOVETTORI]** Sia V un vettore di interi. Chiamiamo costo di un sottovettore di V la somma dei suoi elementi. Sia data una partizione in sottovettori del vettore V, chiamiamo costo della partizione il costo massimo dei suoi sottovettori.
Dato un vettore V di n interi e un intero k con $2 \leq k \leq n$, trovare una partizione di V in esattamente k sottovettori di costo minimo.

    a. Descrivere un algoritmo che risolve il problema nel caso particolare $k = 2$ in $\mathrm{O}(n)$.

    b. Descrivere un algoritmo che risolve il problema nel caso particolare $k = 3$ in $\mathrm{O}(n^2)$.

    c. Descrivere un algoritmo che risolve il problema nel caso generale in $\mathrm{O}(k \cdot n^2)$.

26. **[SOMMA-SOTTOMATRICE]** Presa una matrice quadrata A di $n \times n$ interi, si vuole trovare una un sottomatrice rettangolare la somma dei cui elementi sia massima. La soluzione può essere rappresentata dando le coordinate della prima cella in alto a sinistra e dell'ultima cella in basso a destra della sottomatrice stessa)

    a. Provare che, grazie ad un preprocessing, con il metodo della ricerca esaustiva è possibile risolvere il problema in $O(n^4)$.

    b. Progettare un algoritmo che risolve il problema in $O(n^3)$.

27. [**FANTACALCIO**] Il signor Sensotti è presidente di una grossa squadra di calcio che ha avuto mandato di formare una squadra in grado di vincere il campionato del prossimo anno. La società gli ha messo a disposizione un budget di M fantamiliardi. Sensotti deve scegliere t calciatori tra un insieme di n calciatori $\{a_1, a_2, \ldots a_n\}$ disponibili, dove ogni calciatore a_i ha una quotazione c_i ed un coefficiente di bravura d_i.

    a. Progettare un algoritmo che aiuti il presidente Sensotti a selezionare la squadra migliore possibile con il budget a disposizione in $O(M \cdot t \cdot n)$.

    b. Supponendo che ad ogni calciatore sia associato un codice r_i che identifica un ruolo e che i ruoli possibili sono m, modificare l'algoritmo in modo che la squadra selezionata sia la migliore possibile tra tutte quelle che hanno almeno n_j calciatori di ruolo j, per $j = 1, 2, \ldots m$. Il nuovo algoritmo deve avere complessità $O(N \cdot M \cdot t \cdot n)$ dove $N = \sum_{i=1}^{m} n_i$.

28. [**BI-COLORAZIONE DI ALBERI**] Sia T un albero pesato di n nodi. Una bi-colorazione dell'albero T consiste nell'assegnare un colore bianco o nero a ciascun nodo di T. A seguito di una colorazione il nodo v di T avrà peso $w_0(v)$ se colorato di bianco, $w_1(v)$ altrimenti. Definiamo il peso di una colorazione come la somma dei pesi risultanti dei nodi di T.

    Una colorazione di T è lecita quando non vi è alcun nodo nero che abbia figli di colore nero Dare un algoritmo che trova una colorazione lecita di T di peso massimo in $O(n)$.

    Nel seguito assumiamo che i nodi dell'albero siano etichettati $\{1, 2, \ldots n\}$ e che l'etichettamento sia il risultato di una visita inorder dell'albero. Nota che stando così le cose i nodi figli di i avranno etichetta inferiore ad i ed il nodo radice avrà etichetta n.

29. [**FESTA AZIENDALE**] Il Dott. Rossi, presidente della Macroloft, vuole organizzare una festa aziendale ed ha dato mandato all'ufficio personale di selezionare gli invitati. L'organigramma della compagnia ha una struttura gerarchica (ad albero) dove il presidente è la radice ed ogni dipendente è un nodo che ha come figli i nodi corrispondenti ai dipendenti che sono direttamente sotto il suo controllo. Per la buona riuscita della festa il Dott. Rossi ha deciso che se Tizio è il capo diretto di Caio allora uno di questi due dipendenti non deve essere invitato alla festa. L'ufficio del personale ha classificato tutti i dipendenti assegnando ad essi un coefficiente di simpatia s_i.

    a. Dare un algoritmo che seleziona una lista di invitati in modo da rispettare le regole fissate dal Dott. Rossi e da massimizzare la somma dei coefficienti di simpatia degli invitati in $O(n)$ dove n è il numero dei dipendenti.

    b. Modificare l'algoritmo in modo da garantire che il Dott. Rossi sia tra gli invitati e che gli invitati siano al più K.

30. [**OSPEDALI**] C'è una strada che collega n città. La strada può essere pensata come un'asse, la posizione di ciascuna città è pertanto identificata da una singola coordinata e la distanza fra due città è il valore assoluto della differenza delle loro coordinate. Disponiamo di risorse sufficienti alla costruzione di k ospedali, $k < n$. Bisogna individuare le k città in cui costruire gli ospedali in modo che la somma totale di tutte le distanze fra ciascuna città e il più vicino ospedale risulti minima.

    In altri termini il problema da risolvere è il seguente: Dato un insieme $X = \{x_1, x_2 \ldots x_n\}$ di interi positivi ed un intero $k < n$. Il costo di un sottoinsieme S di $\{1, 2, \ldots, n\}$ è definito come

$\sum_{i=1}^{n} d_S(x_i)$ dove $d_S(x_i) = \min_{j \in S}\{|x_i - x_j|\}$. Vogliamo individuare il sottoinsieme S di costo minimo.

Ad esempio per $X = \{1, 4, 6, 8, 9, 10\}$ e $k = 2$ il sottoinsieme di costo minimo è $\{2, 5\}$ ed ha costo 8.

a. Descrivere un algoritmo che risolve il problema nel caso particolare $k = 2$ in $\mathrm{O}(n)$.

b. * Descrivere un algoritmo che risolve il problema nel caso generale in $\mathrm{O}(k \cdot n^3)$.

31. **[AEREOPORTO]** Bisogna costruire un aereoporto e si dispone di una mappa H della zona in cui va localizzato. La mappa suddivide il territorio in una griglia quadrata di $n \times n$ quadrati unitari, e per ogni quadrato unitario viene indicata l'altezza corrispondente. Bisogna localizzare la regione rettangolare di area massima (vale a dire la regione rettangolare che consiste nel maggior numero di quadrati) tale che il *dislivello* all'interno della regione non superi un dato limite C (i.e. la differenza d'altezza tra il quadrato di altezza massima della regione selezionata e quello d'altezza minima della regione selezionata deve essere al più C). Descrivere un algoritmo che presa la mappa H, e l'intero $C << n$ con complessità $O(C \cdot n^3)$ restituisce una regione richiesta (la regione da restituire può essere rappresentata tramite le coordinate della prima cella in alto a sinistra e le coordinate dell'ultima cella in basso a destra della regione stessa). Ad esempio per la mappa 8×8

20	18	21	21	20	18	23	20
19	17	20	21	16	18	20	21
18	19	|20	21	18	18	20	20|
21	17	|18	18	21	21	19	19|
20	21	|18	20	20	21	19	18|
18	20	|19	18	21	20	18	21|
18	20	|18	20	20	21	21	20|
22	18	18	18	22	21	22	21

con $C = 4$ una soluzione è il rettangolo di lato 5 individuato dalle coordinate $(3, 3)$ e $(7, 8)$ ed evidenziato in figura.

32. **[DEPOSITI E COMMERCIANTI]** Ci sono due depositi dai quali prelevare merce per soddisfare le richieste di n commercianti. Siano $d_1, d_2, \cdots d_n$ le quantità di merce richieste dagli n commercianti e, posto d come somma delle quantità richieste, sia C una matrice $2 \times n \times d$ dove $C[i, j, x]$ è il costo per il trasporto di x unità di merce dal deposito i al commerciante j ($c[i, j, x] = +\infty$ se non è possibile trasportare x unità di merce da i a j). Progettare un algoritmo che calcoli il costo minimo per soddisfare le n richieste ($+\infty$ se ciò non è possibile). Nel caso in cui il costo minimo sia finito, l'algoritmo deve poi produrre una matrice X di dimensione $2 \times n$ dove $X[i, j]$ contiene la quantità di merce che dal deposito i va trasportata al commerciante j per garantire che il costo totale sia minimo. Valutare la complessità dell'algoritmo.