ESERCIZI SULLA TECNICA *BACKTRACKING* e *BRANCH & BOUND*

1. [**STRINGHE**] Scrivere in pseudo-codice una procedura che, preso in input un intero n, stampi tutte le stringhe di lunghezza minore o uguale ad n sull'alfabeto $\{a, b, c\}$.

2. [**SOTTOINSIEMI CON PARI**] Scrivere in pseudo-codice una procedura che, preso in input un intero n, stampi tutti i sottinsiemi di $\{1, 2, \ldots, n\}$ che contengono almeno un elemento che è un numero pari. La complessità del programma deve essere $O(nD(n))$, dove $D(n)$ è il numero di stringhe da stampare.

3. [**STRINGHE BINARIE BILANCIATE**] Scrivere in pseudo-codice una procedura che, preso in input un intero n, stampi tutte le stringhe di lunghezza $2n$ sull'alfabeto $\{a, b\}$ con lo stesso numero di occorrenze dei simboli a e b. La complessità del programma deve essere $O(nD(n))$, dove $D(n)$ è il numero di stringhe da stampare.

4. [**STRINGHE TERNARIE SEMIBILANCIATE**] Scrivere in pseudo-codice una procedura che, preso in input un intero n, stampi tutte le stringhe di lunghezza n sull'alfabeto $\{a, b, c\}$ con lo stesso numero di occorrenze dei simboli a e b. La complessità del programma deve essere $O(nD(n))$, dove $D(n)$ è il numero di stringhe da stampare.

5. [**SEQUENZE CRESCENTI**] Scrivere in pseudo-codice una procedura che, preso in input un intero n, stampi tutte le sequenze strettamente crescenti di lunghezza n sull'alfabeto $\{1, 2, \ldots 2{\cdot}n\}$. Ad esempio se $n = 2$ allora il programma deve stampare (non necessariamente in quest'ordine): $(1, 2)$, $(1, 3)$, $(1, 4)$, $(2, 3)$, $(2, 4)$ e $(3, 4)$. La complessità del programma deve essere $O(nD(n))$, dove $D(n)$ è il numero di stringhe da stampare.

6. [**SEQUENZE PARI-DISPARI**] Scrivere in pseudo-codice una procedura che, presi in input due interi n e k stampi tutte le sequenze di lunghezza n con elementi in $\{0, 1, \ldots, k\}$ in cui non compaiono né due numeri pari consecutivi né due numeri dispari consecutivi. Ad esempio se $n = 3$ e $k = 2$ allora il programma deve stampare (non necessariamente in quest'ordine): $(0, 1, 0)$, $(0, 1, 2)$, $(1, 0, 1)$, $(1, 2, 1)$, $(2, 1, 0)$ e $(2, 1, 2)$. La complessità del programma deve essere $O(nD(n))$, dove $D(n)$ è il numero di stringhe da stampare.

7. [**SEQUENZE ZIG-ZAG**] Scrivere in pseudo-codice una procedura che, preso in input un intero n, stampi tutte le sequenze di lunghezza n con elementi in $\{0, 1, 2\}$ in cui per qualsiasi terna x_i, x_{i+1}, x_{i+2} di numeri consecutivi risulta $x_i < x_{i+1}$ e $x_{i+1} > x_{i+2}$ oppure $x_i > x_{i+1}$ e $x_{i+1} < x_{i+2}$. Ad esempio se $n = 3$ allora il programma deve stampare (non necessariamente in quest'ordine): $(0, 1, 0)$, $(0, 2, 0)$, $(0, 2, 1)$, $(1, 0, 1)$, $(1, 0, 2)$ e $(1, 2, 0)$, $(1, 2, 1)$, $(2, 0, 1)$, $(2, 0, 2)$ e

$(2, 1, 2)$. La complessità del programma deve essere $O(nD(n))$, dove $D(n)$ è il numero di stringhe da stampare.

8. [**SOTTOINSIEMI QUASI DISPARI**] Scrivere in pseudo–codice una procedura che, preso in input un intero n, stampi tutti i sottoinsiemi di $\{1, 2, \ldots, n\}$ che contengono al più un elemento che sia un numero pari. Ad esempio se $n = 4$, la procedura deve stampare (non necessariamente in quest'ordine) i seguenti sottoinsiemi: $\emptyset$, $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{1, 2\}$, $\{1, 3\}$, $\{1, 4\}$, $\{2, 3\}$, $\{3, 4\}$, $\{1, 2, 3\}$ e $\{1, 3, 4\}$. La complessità del programma deve essere $\mathrm{O}(nD(n))$, dove $D(n)$ è il numero di sottoinsiemi da stampare.

9. [**SOTTOINSIEMI DI EGUAL VALORE**] Scrivere in pseudo-codice una procedura che, presi in input due interi n e k, stampi tutti i sottinsiemi di $\{1, 2, \ldots, n\}$ la somma dei cui elementi sia k. Ad esempio se $n = 5$ e $k = 7$ allora il programma deve stampare (non necessariamente in quest'ordine): $\{1, 2, 4\}$, $\{2, 5\}$, $\{3, 4\}$.

10. [**SOTTOINSIEMI MINIMO = CARDINALITÀ**] Scrivere in pseudo–codice una procedura che, preso in input un intero positivo n, stampi tutti i sottoinsiemi di $\{1, 2, \ldots, n\}$ il cui minimo è uguale alla cardinalità. Ad esempio se $n = 6$ la procedura deve stampare (non necessariamente in quest'ordine) i seguenti sottoinsiemi: $\{1\}$, $\{2, 3\}$, $\{2, 4\}$, $\{2, 5\}$, $\{2, 6\}$, $\{3, 4, 5\}$, $\{3, 4, 6\}$, $\{3, 5, 6\}$. La complessità del programma deve essere $\mathrm{O}(nD(n))$, dove $D(n)$ è il numero di sottoinsiemi da stampare.

11. [**SEQUENZE ABC**] Scrivere in pseudo-codice una procedura che, preso in input un intero n, stampi tutte le sequenze di lunghezza n con elementi in $\{a, b, c\}$ dove il numero di occorrenze di a è minore o uguale al numero di occorrenze di b che a sua volta è minore o uguale al numero di occorrenze di c. Ad esempio se $n = 3$ allora il programma deve stampare (non necessariamente in quest'ordine): $\{c, c, c\}$, $\{b, c, c\}$, $\{c, b, c\}$, $\{c, c, b\}$, $\{a, b, c\}$, $\{b, a, c\}$, $\{b, c, a\}$, $\{a, c, b\}$, $\{c, a, b\}$ e $\{c, b, a\}$. La complessità del programma deve essere $O(nD(n))$, dove $D(n)$ è il numero di stringhe da stampare.

12. [**COMBINAZIONE**] Sappiamo che la combinazione di una cassaforte è composta da una sequenza di n cifre decimali la cui somma ha valore k. Scrivere in pseudo-codice una procedura che, dati n e k, produce tutte le possibili combinazioni con questa proprietà. Il programma deve avere complessità $O(nC(n))$, dove $C(n)$ è il numero di combinazioni possibili.

13. [**CRICCHE**] Si consideri un grafo non diretto $G = (V, E)$. Una *cricca* di G è un sottoinsieme V' di V tale che tutti i vertici di V' sono tra loro adiacenti. Una k-cricca di G, $1 \leq k \leq |V|$ è una cricca del grafo con k elementi. Una cricca è massimale se non è sottoinsieme di altre cricche.

    a. Scrivere in pseudo-codice una procedura che, presi in input un intero k ed un grafo non diretto G, stampi tutte le k-cricche di G.

    b. Scrivere in pseudo-codice una procedura che, presi input un grafo non diretto G, stampi tutte le cricche massimali.

14. [**ISOMORFISMO DI GRAFI**] Due grafi $G = (V, E)$ e $H = (A, B)$ sono *isomorfi* se esite una corrispondenza uno-a-uno dei vertici che preserva le relazioni di adiacenza. Più formalmente G e H sono isomorfi se esiste una funzione $f : V \Rightarrow A$ surriettiva tale che per ogni $x, y \in V$ si ha

$\{x, y\} \in E$ se e solo se $\{f(x), f(y)\} \in B$. Scrivere in pseudo-codice una procedura che, presi in input due grafi, stampa tutti i possibili isomorfismi.

15. [**k-COLORAZIONI**] Dato un intero k ed un grafo non diretto G, una *k-colorazione* di G è un'assegnazione di un colore in $\{1, 2, \ldots, k\}$ ad ogni vertice di G in modo tale che vertici adiacenti abbiano colori distinti. Scrivere in pseudo-codice una procedura che, presi in input un intero k e un grafo non diretto G, stampi tutte le k-colorazioni di G (una k-colorazione si può rappresentare tramite un array $sol[1 : n]$ dove $sol[i]$ è il colore assegnato al vertice i).

16. [**REGINE**] Su di una scacchiera $n \times n$ due regine non si contrastano se sono posizionate su caselle appartenenti a righe diverse, colonne diverse e diverse diagonali. Un'assegnamento di n regine ad n caselle della scacchiera di modo che queste non si contrastino determina una configuarazione lecita.

    a. Scrivere in pseudo-codice una procedura che, preso in input un intero n, stampi tutte le configurazioni lecite per la scacchiera $n \times n$. (In una configurazione lecita 2 delle n regine non possono essere sulla stessa riga della scacchiera, una configurazione lecita si può quindi rappresentare tramite un vettore $sol[1 : n]$ dove $sol[i]$ è la colonna in cui si trova la regina posizionata nell'i-esima riga).

    b. Scrivere in pseudo-codice una procedura che, preso in input un intero n, determini il numero minimo di regine da posizionare sulla scacchiera $n \times n$ perchè tutte le caselle non occupate risultino "sotto attacco" da almeno una regina.

17. [**PERMUTAZIONI**] Scrivere in pseudo-codice una procedura che, preso in input un intero n, stampi tutte le $n!$ permutazioni degli elementi $\{1, 2, \ldots, n\}$. Il programma deve avere complessità $O(n \cdot n!)$,

18. [**CAMMINI HAMILTONIANI**] Un *cammino hamiltoniano* in un grafo diretto G è un cammino che tocca ogni vertice del grafo esattamente una volta. Un ciclo hamiltoniano di G è un ciclo che passa attraverso ogni vertice del grafo esattamente una volta.

    a. Scrivere in pseudo-codice una procedura che, preso in input un grafo diretto G, stampi tutti i cammini hamiltoniani del grafo G (un cammino hamiltoniano si può rappresentare tramite un array $sol[1 : n]$ dove $sol[i]$ è l'etichetta dell'i-esimo vertice del cammino).

    b. Modificare l'algoritmo del punto *a.* in modo che stampi i diversi cicli hamiltoniani. (Un ciclo hamiltoniano si può rappresentare tramite un array $sol[1 : |V|]$ dove $sol[i + 1]$ è l'etichetta del vertice che nel ciclo segue il vertice con etichetta $sol[i]$, per $1 \leq i < |V|$, mentre $sol[1]$ è l'etichetta del vertice che nel ciclo segue il vertice con etichetta $sol[|V|]$).

19. [**TOUR DEL CAVALLO**] Si consideri un cavallo posto sulla casella (x, y) di una ipotetica scacchiera $n \times n$. Il problema richiede di determinare le $n^2 - 1$ mosse da fare perchè il cavallo tocchi tutte le caselle della scacchiera una ed una sola volta (posto che una tale sequenza di mosse esista). Scrivere in pseudo-codice una procedura che, presi n e (x, y), risolve questo problema. Inoltre descrivere un algoritmo che trova, se esiste, una sequenza di mosse che oltre ad avere le proprietà suddette ritorna alla casella di partenza.

20. [**QUADRATI LATINI**] Un *quadrato latino* di ordine n è una matrice $n \times n$ contenente gli interi $\{1, 2, \ldots, n^2\}$ assegnati in modo tale che ogni riga ed ogni colonna della matrice è una permutazione di $\{1, 2, \ldots, n^2\}$. Scrivere in pseudo-codice una procedura che, dato n, stampa i diversi quadrati latini in cui gli lementi della prima riga e della prima colonna occorrono nell'ordine naturale $\{1, 2, \ldots, n^2\}$.

21. [**QUADRATI MAGICI**] Un *quadrato magico* di ordine n è una matrice $n \times n$ contenente tutti gli interi in $\{1, 2, \ldots, n^2\}$ assegnati in modo tale che la somma di ogni riga, di ogni colonna e delle due diagonali sia la stessa.

    Ad esempio per $n = 5$ un possibile quadrato magico è

1	15	24	8	17
23	7	16	5	14
20	4	13	22	6
12	21	10	19	3
9	18	2	11	25

    Scrivere in pseudo-codice una procedura che, preso in input un intero n, stampi i diversi quadrati magici di ordine n (nota che la somma di tutti gli elementi della matrice è

    $$\sum_{i=1}^{n^2} i = \frac{n^2(n^2+1)}{2}$$

    quindi la somma comune alle righe, alle colonne ed alle due diagonali deve essere $\frac{n(n^2+1)}{2}$).

22. [**ASSEGNAMENTO**] Ci sono n persone cui vanno assegnati n lavori. Si dispone di una tabella $Tn \times n$ dove nella locazione $T[i, j]$ troviamo il costo di aseegnare l'i-esimo lavoro alla j-ma persona. Scrivere in pseudo-codice una procedura che, dati n e la tabella T, assegna ogni lavoro ad una diversa persona ed allo stesso tempo minimizza il costo totale degli assegnamenti.

23. [**ASSEGNAMENTO1**] Ci sono n lavori che vanno assegnati a k macchine che possono eseguirli in parallelo, $k \leq n$. Si dispone di un vettore V ad n componenti dove nella locazione $V[i]$ troviamo il tempo necessario per l'esecuzione dell'i-esimo lavoro. Descrivere in pseudo-codice una procedura che, presi il vettore V e e gli interi n e k, produce un assegnamento degli n lavori alle k macchine ed allo stesso tempo minimizza il il tempo di fine dell'esecuzione di tutti i lavori.

24. [**TRIPLE**] Siano date $3n$ persone ed una matrice P tridimensionale $3n \times 3n \times 3n$ tale che $P(i, j, k)$ è la preferenza della persona i per le persone j e k. Assumiamo che $P(i, j, k) = P(i, k, j)$ per qualsiasi scelta di i, j, k. Per tre persone i, j, k l'affiatamento della tripla $\{i, j, k\}$ è dato dal prodotto delle preferenze $P(i, j, k)P(j, i, k)P(k, i, j)$. Scrivere in pseudo-codice una procedura che, presi in input n e la matrice P, produce una suddivisione in triple delle $3n$ persone in modo da massimizzare la somma degli affiatamenti.

25. [**VETTORE CIRCOLARE**] Dato un vettore circolare V di n locazioni (vale a dire un vettore in cui la locazione $i \geq 0$ equivale alla locazione $i \bmod n$) contenente interi, si dice che un intero x è generabile da V se esiste un sottovettore di V la somma dei cui elementi è x. Scrivere in pseudo-codice una procedura che, presi in input tre interi positivi n, k ed m, stampa tutti i

distinti vettori circolari di n locazioni i cui elementi sono interi maggiori o uguali a k e tali che la sequenza continua di elementi generati dai vettori e che ha come primo elemento m sia la più lunga possibile. Per evitare di stampare rotazioni di uno stesso vettore circolare si stampi tra tutte le rotazioni quella che presenta l'elemento più piccolo in prima posizione. Ad esempio per $n = 5$, $k = 1$ ed $m = 2$ La sequenza più lunga che parte da 2 e può ottenersi con un vettore circolare di 5 locazioni con elementi maggiori e uguali di 1 va da 2 a 21 e può ottenersi con 4 diversi vettori circolari. Il programma stamperà

1	3	10	2	5
1	5	2	10	3
2	4	9	3	5
2	5	3	9	4

26. [**UFFICIO POSTALE**] L'ufficio postale di uno stato permette l'uso di francobolli di n valori interi e diversi e proibisce l'uso di più di m francobolli per lettera. Scrivere in pseudo-codice una procedura che, presi in input m ed n, calcola il massimo intervallo a partire da 1 di affrancature possibili e tutti gli insiemi di valori che permettono di realizzarlo. Ad esempio, per $n = 4$ e $m = 5$ l'intervallo massimo è $\{1, 2 \ldots, 71\}$ che si può ottenere solamente con i seguenti due insiemi di valori $\{1, 4, 12, 21\}$ e $\{1, 5, 12, 28\}$.

27. [**TASSELLAMENTO**] L'identità $1^2 + 2^2 + 3^2 \ldots + 24^2 = 70^2$ suggerisce che è possibile tassellare un quadrato di area 70×70 usando 24 quadrati di area $1, 2, \ldots, 24$ rispettivamente. Scrivere in pseudo-codice una procedura che verifichi se un tale tassellamento è possibile.