ALGORITMICA Verifica del 21 Dicembre 2005

Cognome Nome: **N.Matricola:** **Corso: B**

Esercizio 1. *(10 punti)* Ipotizzate di avere la sequenza $S = 7, 2, 6, 4, 13, 19, 32, 9$ di chiavi da inserire in una tabella hash di dimensione m.

1. Mostrate il contenuto della tabella supponendo che le collisioni siano gestite con liste concatenate e che la funzione hash sia basata sul metodo della divisione, dove $m = 7$ e $h_1(k) = k \bmod m$.

2. Mostrate il contenuto della tabella supponendo che le collisioni siano gestite con indirizzamento aperto utilizzando l'hash doppio come metodo di scansione con $m = 11$, dove h_1 è definita come sopra e $h_2(k) = [(k \times 7) \bmod 3] + 1$.

Soluzione:

```
k   7  2  6  4 13 19 32  9
h1  0  2  6  4  6  5  4  2

a[0] = {7}
a[1] = null
a[2] = {2, 9}
a[3] = null
a[4] = {4, 32}
a[5] = {19}
a[6] = {6, 13}
```

Usando la funzione h_1 del secondo punto:

```
k   7  2  6  4 13 19 32  9
h1  7  2  6  4  2  8 10  9
h2  2  3  1  2  2  2  3  1

- - 2 - 4 32 6 7 13 9 19
```

Usando la funzione h_1 del primo punto:

```
k   7  2  6  4 13 19 32  9
h1  0  2  6  4  6  5  4  2
h2  2  3  1  2  2  2  3  1

7 - 2 9 4 19 6 32 13 - -
```

ALGORITMICA Verifica del 21 Dicembre 2005

Cognome Nome: **N.Matricola:** **Corso: B**

Esercizio 2. *(8 punti)* Progettate un algoritmo di tipo divide et impera per trovare simultaneamente il minimo e il massimo in un array A di n elementi. Calcolate la complessità temporale dell'algoritmo proposto utilizzando le equazioni di ricorrenza e discutendone la risoluzione.

Soluzione:

```
MinMax(p, r):
if (p > r) return <-infinito, +infinito>;
q = (p+q)/2;
<m1, M1> = MinMax(p, q);
<m2, M2> = MinMax(q+1, r);
m = min( m1, m2);
M = max ( M1, M2);
return < m, M >;
```

Equazione di ricorrenza: $T(n) = 2T(n/2) + O(1)$.
Soluzione: I caso del teorema fondamentale delle ricorrenze con $a = 2, b = 2, f(n) =$ costante. Quindi $T(n) = O(n^{\log_b a}) = O(n)$.

ALGORITMICA Verifica del 21 Dicembre 2005

Cognome Nome: **N.Matricola:** **Corso: B**

Esercizio 3. *(12 punti)* Un grafo non orientato $G = (V, E)$ è dato in ingresso con numero pari $n = |V|$ di vertici.

1. Scrivete lo pseudocodice di un algoritmo che stabilisce se esistono in G due cicli semplici (i vertici intermedi appaiono una sola volta) e disgiunti, dove ciascun ciclo è formato da $n/2$ vertici. Discutetene la complessità temporale, eventualmente confrontandola con quella di problemi simili discussi nel corso.

   Soluzione: È il problema del cammino hamiltoniano in un grafo: basta generare tutte le permutazioni dei vertici memorizzandole in un array: presi i primi $n/2$ vertici della permutazione, verifichiamo che questi formino un ciclo (deve esistere l'arco che collega due vertici consecutivi); presi gli ultimi $n/2$ vertici, verifichiamo che questi formino un ciclo separato dal primo. Ovviamente possono esistere degli archi che connettono i due cicli, ma quelli dei due cicli sono disgiunti. Si noti che duplicando un grafo, possiamo verificare se ammette un ciclo hamiltoniano in questo modo, quindi il problema e' NP-completo e non si conoscono soluzioni polinomiali, come discusso a lezione.

2. Scrivete lo pseudocodice di un algoritmo che ricevuto in ingresso G e un intero $k > 0$, conta il numero di coppie di vertici $u, v \in V$ la cui distanza è minore o uguale a k. Valutate la complessità temporale della soluzione proposta.

   Soluzione: Si consideri BFS(u): la profondità dei nodi nell'albero BFS misura la distanza (lunghezza del cammino minimo) degli altri nodi rispetto a u, in tempo $O(n + m)$. Iterando BFS(u) per tutti i vertici u otteniamo tutte le distanze in tempo $O(n^2 + nm)$: tra queste scegliamo solo quello che sono minori o uguali a k. Un'ottimizzazione consiste nell'interrompere la BFS non appena si raggiunge un nodo di profondità maggiore di k nell'albero BFS.

3. Scrivete lo pseudocodice di un algoritmo che riceve in ingresso G e quattro vertici u, v, w, z e che restituisce il valore TRUE se e solo se esiste un cammino dal vertice u al vertice z, che *passa* attraverso il vertice v ma *evita* il vertice w. Valutate la complessità temporale.

   Soluzione: Si tratta di unire le soluzioni di due esercizi già dati in esami precedenti. La visita (DFS o BFS) a partire da un nodo u ci permette di stabilire quali nodi sono connessi a u (quelli marcati come visti nel vettore B): eseguiamo prima DFS(u) e verifichiamo che possa arrivare a v; poi eseguiamo nuovamente DFS, ma a partire da v e verifichiamo che possa giungere a z. In questo modo siamo certi di poter andare da u a z passando per v. Per evitare di passare da w, basta porre $B[w] =$ TRUE durante l'inizializzazione di B in entrambe le visite DFS. Il costo è $O(n + m)$ come per le visite.