PRIMA PARTE

1. **[4 punti]** Sia A un algoritmo che risolve un problema computazionale Π e la cui complessità al caso pessimo è $\Theta(n^2)$. Giustificare rigorosamente la seguente affermazione: *per ogni n abbastanza grande esiste un'istanza di Π di taglia n per risolvere la quale A esegue $> 1000 \cdot n$ operazioni.*

2. **[6 punti]** Si consideri il pattern P=CONCORSO.

   (a) Determinare la failure function f per P

   (b) Supponendo di calcolare f per P con l'algoritmo efficiente (computeFailKMP), si consideri l'iterazione in cui $i = 5$ e $j = 2$ (si confrontano quindi $P[i] = R$ e $P[j] = N$) e sono stati già calcolati i valori $f(0), f(1), \ldots f(4)$. Dire quali operazioni vengono svolte dall'algoritmo in questa e nella successiva iterazione

3. **[6 punti]** Dimostrare che in un albero binario completo con n nodi e altezza h vale che $h \leq \log_2 n$.

SECONDA PARTE

1. **[7 punti]** Si consideri un algoritmo ricorsivo A che per ogni istanza di taglia n fa quanto segue

   - Se $n = 1, 2, 3$, esegue 10 operazioni;
   - Se $n > 3$, esegue $6n$ operazioni e poi invoca ricorsivamente se stesso su un'istanza di taglia $n - 3$.

   Detta $t_A(n)$ la complessità di A, dimostrare, per induzione su n, che $t_A(n) \geq n^2$ per ogni $n \geq 1$.

2. **[9 punti]** Si vuole progettare un algoritmo *ricorsivo* `heightSum` per calcolare la somma delle altezze di tutti i nodi di un albero binario proprio T. Detta `heightSum(T,v)` la generica invocazione su un nodo $v \in T$, per risolvere il problema si eseguirà `heightSum(T,T.root())`.

   (a) Descrivere tramite pseudocodice `heightSum(T,v)`, specificandone con attenzione l'input e l'output.

   (b) Analizzare la complessità di `heightSum(T,T.root())` in funzione del numero n di nodi in T.

TEMPO COMPLESSIVO A DISPOSIZIONE: 1.5 ore