Ingegneria degli Algoritmi
30 gennaio 2020

Esercizio 1
Trovare un limite superiore ed un limite inferiore, i più stretti possibili, per la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 2T(n/8) + \sqrt[3]{n} & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Soluzione
Utilizzando il master theorem, è facile vedere che $T(n) = \Theta(\sqrt[3]{n}\ logn)$.
Infatti, data l'equazione di ricorrenza parametrica:

$$T(n) = \begin{cases} aT(n/b) + cn^{\beta} & n > 1 \\ d & n \leq 1 \end{cases}$$

abbiamo $a = 2$, $b = 8$, $c = 1$, $\beta = 1/3$, $d = 1$. Poniamo $\alpha = log_8 2 = 1/3$. Poiché $\alpha = \beta$, ricadiamo nel secondo caso del master theorem, da cui $T(n) = \Theta(n^{\alpha}\ logn) = \Theta(\sqrt[3]{n}\ logn)$.

Esercizio 2
La città di *Hateville* è un villaggio particolare. È composto da una sola strada, con case numerate da 1 ad n, dislocate lungo quella sola strada. Ad Hateville, ognuno odia i propri vicini della porta accanto, da entrambi i lati. Quindi, il vicino della porta i odia i vicini delle porte $i-1$ ed $i+1$ (se esistenti).
Hateville vuole organizzare una sagra e vi ha affidato il compito di raccogliere i fondi. Ogni abitante nella casa i ha intenzione di donare una quantità $D[i]$, ma non intende partecipare ad una raccolta fondi a cui partecipano uno o entrambi i propri vicini.
Scrivere un algoritmo (in pseudocodice o in python) che restituisca la quantità massima di fondi che può essere raccolta. Discutere (informalmente) della complessità associata all'algoritmo proposto.

Soluzione
È possibile definire una formula ricorsiva che ci permetta di calcolare il sottoinsieme di case che, se selezionate, dà origine alla maggior quantità di donazioni. Per fare questo, ridefiniamo il problema:

- Sia $HV(i)$ uno dei possibili insiemi di indici da selezionare per ottenere una donazione ottimale dalle prime i case di Hateville, numerate $1 \ldots n$;
- $H(n)$ è la soluzione del problema originale.

Considerate il vicino i-esimo:

- Cosa succede se non accetto la sua donazione?:
  $HV(i) = HV(i-1)$
- Cosa succede se accetto la sua donazione?
  $HV(i) = \{i\} \cup HV(i-2)$
- Come faccio a decidere se accettare o meno?
  $HV(i) = max\{HV(i-1),\ \{i\} \cup HV(i-2)\}$

A questo punto possiamo completare la ricorsione:

- $HV(0) = \varnothing$

- $HV(1) = 1$

e mettendo tutto insieme:

$$HV(i) = \begin{cases} \emptyset & i = 0 \\ \{1\} & i = 1 \\ highest(HV(i-1), HV(i-2) \cup \{i\}) & i \geq 2 \end{cases}$$

Questa formulazione si presta bene all'utilizzo della tecnica della programmazione dinamica, dando luogo al seguente pseudocodice:

```
Hateville(D, n):
  DP ← [0] * (n)
  DP[0] ← 0
  DP[1] ← D[1]
  for i ← 2 to n:
     DP[i] ← max(DP[i - 1], DP[i - 2] + D[i])
  return DP[n]
```

Una possibile implementazione in python è la seguente:

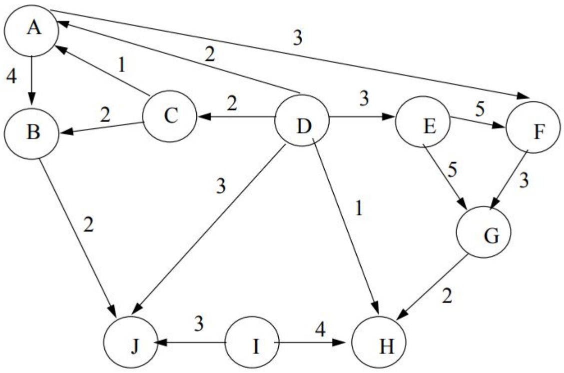
```
def hateville(D):
  DP = [ 0, D[0] ]
  for i in range(1, len(D)):
     DP.append( max(DP[-1], DP[-2] + D[i]))
  return DP[-1]
```

Una possibile soluzione ricorsiva, anziché iterativa, può essere la seguente, che nasce direttamente dalla definizione ricorsiva del problema:

```
def hateville_r(D, i):
    if i == 0:
        return 0
    if i == 1:
        return D[0]
    return max(hateville_r(D, i-1), hateville_r(D, i-2)+D[i-1])
```

che può essere invocata come `hateville_rec(D, len(D))`. Chiaramente, questa soluzione è meno efficiente, poiché (come discusso durante il corso in relazione al calcolo dell'i-esimo numero di Fibonacci) richiede di calcolare più volte la stessa sequenza di valori.

Esercizio 3
Sia dato il seguente grafo orientato pesato:



Si determinino i valori di tutti i cammini minimi che collegano il vertice D con ogni altro vertice mediante l'algoritmo di Dijkstra, fornendo prima una descrizione (in pseudocodice) dell'algoritmo.

Soluzione
Valori finali:

A	B	C	D	E	F	G	H	I	J
2	4	2	0	3	5	8	1	∞	3