Problem Set 2

docente: Luciano Gualà

Esercizio 1 (*equazioni di ricorrenza*)
Si risolvano le seguenti equazioni di ricorrenza. Si assuma sempre $T(1) = 1$.

(*a*) $T(n) = T(n - 10) + 10$.

(*b*) $T(n) = T(n/2) + 2^n$.

(*c*) $T(n) = T(n/3) + T(n/6) + n^{\sqrt{\log n}}$.

(*d*) $T(n) = T(\sqrt{n}) + \Theta(\log \log n)$.

(*e*) $T(n) = T(n/2 + \sqrt{n}) + \Theta(1)$.

(*f*) $T(n) = \sqrt{n}T(\sqrt{n}) + n$.

Esercizio 2 (*test di infrangibilità di bicchieri*)
Dovete valutare l'infrangibilità di un certo tipo di bicchiere di vetro e organizzate un test per determinare l'altezza massima da cui potete far cadere il bicchiere senza che esso si rompa. Per il test, avete a disposizione una scala con n pioli e voi dovete capire quale è il più alto piolo da cui è possibile far cadere il bicchiere senza conseguenze. Chiameremo questo piolo il *più alto piolo sicuro*.

Ora, essendo dei bravi algoritmisti, se aveste a disposizione tante copie dello stesso tipo di bicchiere, al fine di fare pochi lanci, simulereste una ricerca binaria. Quindi lascereste cadere un bicchiere dal piolo di altezza $n/2$ e, in base all'esito, passereste a provare il piolo ad altezza $n/4$ o $3n/4$, e così via. Questo vi consentirebbe di trovare il più alto piolo sicuro facendo $O(\log n)$ lanci ma potreste rompere molti bicchieri.

Se invece il vostro obiettivo primario fosse quello di conservare quanti più bicchieri potete, allora la strategia migliore sarebbe quella di provare in sequenza il piolo 1, poi il piolo 2, e così via, fino a trovare il più alto piolo sicuro. Questo sacrificherebbe un solo bicchiere ma vi costerebbe in termini di tempo, perché nel caso peggiore potreste dover eseguire un numero lineare di lanci.

In questo esercizio vi si chiede di trovare una soluzione che bilancia il numero di lanci con il numero di bicchieri che potete rompere. Per essere più precisi, considerate il caso in cui avete a disposizione un numero massimo $k \geq 1$ di bicchieri che potete rompere, e voi volete capire quale è il più alto piolo sicuro effettuando meno lanci possibile. In particolare:

(a) Supponete di avere a disposizione $k = 2$ bicchieri. Trovate una strategia che risolva il problema eseguendo al più $f(n)$ lanci, per una qualche funzione $f(n) = o(n)$.

(b) Ora assumete di avere a disposizione $k > 2$ bicchieri. Descrivete una strategia per trovare velocemente il più alto piolo sicuro utilizzando al più k bicchieri. Se $f_k(n)$ denota il numero di lanci che occorrono alla vostra strategia per un certo k, allora le funzioni $f_1(n), f_2(n), \ldots$ dovrebbero avere la caratteristica di possedere un tasso di crescita via via più basso al crescere di k, ovvero, dovrebbe valere che $f_k(n) = o(f_{k-1}(n))$, per ogni k.

Esercizio 3 (*calcolo della maggioranza*)
Sia dato un vettore $V[1;n]$ di n elementi. Un elemento è di *maggioranza* se il numero di occorrenze dell'elemento è strettamente più di $n/2$. Si assuma che nel vettore c'è un elemento di maggioranza (che si vuole scoprire) ma gli elementi sono in qualche modo "nascosti", non possono essere letti direttamente. L'unica operazione possibile è fare una *richiesta* $q(i,j)$, che consiste nel chiedere se i due elementi in posizione i e j, rispettivamente, sono uguali o meno; ovvero, $q(i,j)$ restituisce `vero` se $V[i] = V[j]$, `falso` altrimenti. Le operazioni di richiesta sono costose e quindi se ne vogliono fare il meno possibile.

(a) Si progetti un algoritmo che usa la tecnica del *divide et impera* che trova l'elemento di maggioranza facendo $o(n^2)$ richieste.

(b) Si progetti un (miglior) algoritmo che fa un numero lineare di richieste e usa memoria costante.

Esercizio 4 (*embedding di un albero binario completo su una griglia bidimensionale*)
Sia dato un albero binario completo T con n foglie e una griglia bidimensionale (foglio a quadretti). Si vuole disegnare T sulla griglia in modo da non far intrecciare gli archi e minimizzando lo spazio utilizzato. Più precisamente, un *embedding* di T è un disegno di T sul foglio tale che: (i) i nodi di T sono disegnati sulle intersezioni della griglia (ovvero come cerchi centrati su un qualche spigolo di un qualche quadratino), (ii) gli archi sono delle linee "seghettate" che seguono le linee del foglio, (iii) le linee che rappresentano gli archi non possono incrociarsi reciprocamente. Dato un embedding di T il suo *bounding box* è il rettangolo più piccolo che contiene l'intero embedding. Trovare un embedding di T che minimizza (asintoticamente) l'area del bounding box.