Cognoms **Nom** **DNI**

Examen Parcial EDA **Duració: 2h45min** **11/11/2019**

- *L'enunciat té 7 fulls, 14 cares i 4 problemes.*
- *Poseu el vostre nom complet i número de DNI a cada full.*
- *Contesteu tots els problemes en el propi full de l'enunciat i a l'espai reservat.*
- *A no ser que es digui el contrari,* **cal justificar totes les respostes**.
- *Sempre que parlem de cost, entendrem* **cost asimptòtic en temps**.

Problema 1 **(1 punt)**

(a) (0.5 pts.) La solució de la recurrència $T(n) = 2T(n/4) + \Theta(\sqrt{n})$ és asimptòticament $T(n) = \Theta($ $)$. No cal que justifiqueu la resposta.

(b) (0.5 pts.) Per a quines $X \in \{O, \Omega, \Theta\}$ es compleix que $\log_2(n) \in X(\log_2(\log_2(n^2)))$?

Aquesta cara estaria en blanc intencionadament si no fos per aquesta nota.

Cognoms **Nom** **DNI**

Problema 2 **(2.5 punts)**

Donat un natural $n \geq 1$, qualsevol funció $f : \{0, 1, \ldots, n-1\} \longrightarrow \{0, 1, \ldots, n-1\}$ es pot representar com un vector d'enters $[f(0), f(1), ..., f(n-1)]$.

Per exemple, si $n = 5$ i $f(0) = 2, f(1) = 1, f(2) = 2, f(3) = 4, f(4) = 3$, la funció f es pot representar pel vector $[2, 1, 2, 4, 3]$. En tot aquest problema, assumirem aquesta representació de funcions.

(a) (0.75 pts.) Considereu el codi següent:

```
void misteri_aux (const vector<int>& f, const vector<int>& g,
                  int i, vector<int>& r) {
  if (i < f.size ()) {
    r[i] = f[g[i]];
    misteri_aux (f,g,i+1,r);
  }
}

vector<int> misteri(const vector<int>& f, const vector<int>& g) {
  // Precondició: f i g tenen la mateix mida i contenen nombres
  // entre 0 i f.size() − 1
  vector<int> r(f.size ());
  misteri_aux (f,g,0,r);
  return r;
}
```

Què retorna la funció *misteri*? No cal que justifiqueu la resposta.

Si assumim que n és la mida de f, quin és el cost de *misteri* en funció de n?

(b) (0.75 pts.) Considereu ara el codi següent:

```
vector<int> misteri_2(const vector<int>& f, int k) {
  if (k == 0) {
    vector<int> r(f.size());
    for (int i = 0; i < f.size(); ++i) r[i] = i;
    return r;
  }
  else return misteri(f, misteri_2(f,k-1));
}
```

Què retorna la funció *misteri*_2? No cal que justifiqueu la resposta.

Quin és el cost de *misteri*_2 en funció només de k?

Cognoms **Nom** **DNI**

(c) (1 pt.) Completeu la funció següent per tal que calculi el mateix que *misteri_2* però sigui més eficient asimptòticament. Analitzeu-ne el cost en funció de k.

```
vector<int> misteri_2_quick(const vector<int>& f, int k) {
  if (k == 0) {
    vector<int> r(f.size());
    for (int i = 0; i < f.size(); ++i) r[i] = i;
    return r;
  }

}
```

Anàlisi del cost en funció de k:

Aquesta cara estaria en blanc intencionadament si no fos per aquesta nota.

Cognoms **Nom** **DNI**

Problema 3 **(3.25 punts)**

Donat un conjunt S de $m = 2n$ enters diferents, volem agrupar-los en parelles de manera que la suma dels seus productes sigui màxima. És a dir, busquem la màxima expressió de la forma $x_0 * x_1 + x_2 * x_3 + \ldots + x_{2n-2} * x_{2n-1}$, on el x_i's són tots els elements de S.

Per exemple, si $S = \{5, 6, 1, 3, 8, 4\}$, dues possibles expressions són $1*5 + 6*3 + 4*8$, que suma 55, i $5*4 + 1*8 + 3*6$, que suma 46. D'entre aquestes dues preferim la primera, tot i que encara n'hi ha d'altres de millors.

La funció *max_suma* calcula la màxima suma de productes de S:

```
int pos_max (const vector<int>& v, int l, int r) {
  int p = l;
  for (int j = l + 1; j ≤ r; ++j)
    if (v[j] > v[p]) p = j;
  return p;
}

int max_suma (vector<int>& S) {
  int suma = 0;
  int m = S.size ();
  for (int i = 0; i < m; ++i) {
    int p = pos_max(S,i,m−1);
    swap(S[i],S[p]);
    if (i%2 == 1) suma += S[i−1]*S[i];
  }
  return suma;
}
```

(a) (1 pt.) Analitzeu el cost en cas pitjor de *max_suma* en funció de m, el nombre d'elements del vector S.

(b) (1 pt.) Expliqueu a alt nivell com implementaríeu una funció que solucionés el mateix problema però que, en el cas pitjor, fos més eficient asimptòticament que *max_suma*. Indiqueu clarament quin és el cost resultant.

(c) (1.25 pts.) Demostreu que la funció *max_suma* retorna la suma de productes màxima.

Ajuda: demostreu primer que si x_0 i x_1 son els dos nombres més grans de S, aleshores una expressió que conté els productes $x_0 * y$ i $x_1 * z$, per certs $y, z \in S$ no pot ser màxima. A continuació utilitzeu aquest fet per demostrar la correctesa de *max_suma* per inducció sobre m.

Cognoms **Nom** **DNI**

Aquesta cara estaria en blanc intencionadament si no fos per aquesta nota.

Cognoms **Nom** **DNI**

Problema 4 **(3.25 punts)**

Durant els propers $n \geq 3$ dies, se celebrarà un important esdeveniment esportiu, pel qual existeix un enorme mercat de compra-venda d'entrades del que ens en volem aprofitar. Sabem que cada dia podrem comprar o vendre una entrada, i també sabem el preu de les entrades en cada dia, donat com una seqüència $(p_0, p_1, \ldots, p_{n-1})$.

(a) (1.25 pts.) Ens assabentem que la seqüència de preus segueix una forma ben particular. Hi ha un únic dia $0 \leq d \leq n-1$ amb preu mínim p_d i sabem que $p_1 > p_2 > \cdots > p_d$ i $p_d < p_{d+1} < \cdots < p_{n-1}$.

El nostre objectiu és comprar una entrada el dia c i vendre-la el dia v, amb $0 \leq c \leq v \leq n-1$ de manera que maximitzem els nostres guanys. És a dir, volem que $p_v - p_c$ sigui màxim. A tal efecte, ompliu els buits del codi següent per tal que la funció *max_guany* retorni aquest parell $< c, v >$ en temps $\Theta(\log n)$ i analitzeu per què la funció resultant té aquest cost.

```
int f(const vector<int>& p, int l, int r){
    if (l + 1 ≥ r) return (p[l] ≤ p[r] ? l : r);
    else {
        int m = (l+r)/2;
        if ( ________ ) return f(p, ____ , ____ );
        else if ( ________ ) return f(p, ____ , ____ );
        else return m;
    }
}

pair<int,int> max_guany (const vector<int>& p) {
    return { ________ , ________ };
}
```

Nota: recordeu que l'expressió $(B \; ? \; E_T : E_F)$ equival a E_T si l'expressió booleana B és certa i equival a E_F altrament.

Anàlisi del cost:

(b) (1 pt.) En el que resta d'exercici, assumiu que la seqüència p no necessàriament té la forma mencionada a l'apartat anterior, sinó que és una seqüència arbitrària de nombres naturals.

En aquest apartat, donat un dia k en el que necessitem disposar d'una entrada, volem saber quin és el màxim benefici que podem obtenir comprant l'entrada en un cert dia c i venent-la en un cert dia v, però que ens garanteixi tenir l'entrada el dia k. És a dir, no ens val qualsevol parell (c, v) sinó que necessitem que $0 \leq c \leq k \leq v \leq n - 1$. Implementeu una funció amb cost $\Theta(n)$ per calcular aquest benefici.

int *max_guany* (**const** *vector*<**int**>& *p*, **int** *k*) {

}

Cognoms **Nom** **DNI**

(c) (1 pt.) Afrontem finalment el problema general, en el que la seqüència p pot tenir qualsevol forma i volem calcular el màxim guany possible $p_v - p_c$ que correspon a comprar una entrada en el dia c i vendre-la posteriorment en el dia v. Expliqueu a alt nivell com implementaríeu una funció que calculés aquest màxim guany i analitzeu-ne el cost. Solucions amb cost $\Omega(n^2)$ rebran 0 punts.

Ajuda: la funció de l'apartat anterior us pot ser útil per implementar una solució basada en dividir i vèncer.

Aquesta cara estaria en blanc intencionadament si no fos per aquesta nota.