Last name(s) **Name** **ID**

Midterm EDA exam Length: 2h45min 11/11/2019

- *The exam has 7 sheets, 14 sides and 4 problems.*
- *Write your full name and ID on every sheet.*
- *Write your answers to all problems in the exam sheets within the reserved space.*
- *Unless otherwise indicated,* **all your answers must be justified**.
- *Whenever we mention cost, we are referring to* **asymptotic cost in time**.

Problem 1 (1 point)

(a) (0.5 pts.) The solution to the recurrence $T(n) = 2T(n/4) + \Theta(\sqrt{n})$ is asymptotically $T(n) = \Theta($ $)$. You do not have to justify your answer.

(b) (0.5 pts.) For which $X \in \{O, \Omega, \Theta\}$ does it hold that $\log_2(n) \in X(\log_2(\log_2(n^2)))$?

This side would be intentionally blank if it were not for this note.

Last name(s) **Name** **ID**

Problem 2 **(2.5 points)**

Given a natural number $n \geq 1$, any function $f : \{0, 1, \ldots, n-1\} \longrightarrow \{0, 1, \ldots, n-1\}$ can be represented as a vector of integers $[f(0), f(1), ..., f(n-1)]$.

For example, if $n = 5$ and $f(0) = 2, f(1) = 1, f(2) = 2, f(3) = 4, f(4) = 3$, the function f can be represented by the vector $[2, 1, 2, 4, 3]$. In what follows, we will use this representation of functions.

(a) (0.75 pts.) Consider the following code:

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

vector<int> mystery(const vector<int>& f, const vector<int>& g) {
  // Precondition: f and g have the same size and
  // they contain numbers between 0 and f.size() - 1
  vector<int> r(f.size());
  mystery_aux(f,g,0,r);
  return r;
}
```

What does the function *mystery* return? You do not have to justify your answer.

If we denote by n the size of f, what is the cost of *mystery* as a function of n?

(b) (0.75 pts.) Let us now consider the following code:

```
vector<int> mystery_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 mystery(f, mystery_2(f, k-1));
}
```

What does the function *mystery_2* return? You do not have to justify your answer.

What is the cost of *mystery_2* as a function of only k?

Last name(s) **Name** **ID**

(c) (1 pt.) Complete the following function so that it returns the same vector as *mystery_2* but it is asymptotically more efficient. Analyze its cost as a function of k.

```
vector<int> mystery_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;
  }



}
```

Cost analysis as a function of k:

This side would be intentionally blank if it were not for this note.

Last name(s) **Name** **ID**

Problem 3 **(3.25 points)**

Given a set S of $m = 2n$ different integers, we want to put them in pairs so that the sum of their products is maximum. That is, we look for the maximum expression of the form $x_0 * x_1 + x_2 * x_3 + \ldots + x_{2n-2} * x_{2n-1}$, where the x_i's are all the elements of S.

For example, if $S = \{5, 6, 1, 3, 8, 4\}$, two possible expressions are $1*5 + 6*3 + 4*8$, that adds 55, and $5*4 + 1*8 + 3*6$, that adds 46. Between these two expressions, we prefer the first one. However, there are still better expressions.

The function *max_sum* computes the maximum sum of products in 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_sum (vector<int>& S) {
  int sum = 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) sum += S[i−1]*S[i];
  }
  return sum;
}
```

(a) (1 pt.) Analyze the worst-case cost of *max_sum* as a function of m, the number of elements in S.

(b) (1 pt.) Explain at a high level how you would implement a function that solves that same problem but it is asymptotically more efficient than *max_sum*. State precisely which is the resulting cost.

(c) (1.25 pts.) Prove that the function *max_sum* returns the maximum sum of products.

Hint: first prove that if x_0 and x_1 are the two largest elements in S, then an expression that contains the products $x_0 * y$ and $x_1 * z$, for some $y, z \in S$ cannot be maximum. Then, use this fact to prove the correctness of *max_sum* by induction on m.

Last name(s) **Name** **ID**

This side would be intentionally blank if it were not for this note.

Last name(s) **Name** **ID**

Problem 4 **(3.25 points)**

An important multi-sport event will be held during the next $n \geq 3$ days. We know that there exists an important black market buying and selling tickets and we want to take profit from it. We know that we can always buy or sell a ticket and we also know the prices of the tickets for every day, given as a sequence $(p_0, p_1, \ldots, p_{n-1})$.

(a) (1.25 pts.) We have realized that the sequence of prices has a very particular form. There is a unique day $0 \leq d \leq n-1$ with minimum price p_d and we know that $p_1 > p_2 > \cdots > p_d$ and $p_d < p_{d+1} < \cdots < p_{n-1}$.

Our goal is to buy a ticket on day c and sell it on day v with $0 \leq c \leq v \leq n-1$ so that we maximize our profit. That is, we want $p_v - p_c$ to be maximum. Fill in the gaps in the following code so that function max_profit returns the pair $< c, v >$ in time $\Theta(\log n)$ and analyze why the resulting function has this 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_profit (const vector<int>& p) {
  return {           ,           };
}
```

Note: the expression $(B \; ? \; E_T : E_F)$ is equivalent to E_T if the Boolean expression B is true, and is equivalent to E_F otherwise.

Cost analysis:

(b) (1 pt.) From now on, assume that the sequence p does not have the form mentioned in part a), but it is an arbitrary sequence of natural numbers.

Given a day k, in which we need to have a ticket, we want to discover which is the maximum profit we can take by buying the ticket in a certain day c and selling it in a certain day v, but making sure that we have the ticket on day k. That is, not every pair (c, v) is valid, we need that $0 \leq c \leq k \leq v \leq n-1$. Implement a function with cost $\Theta(n)$ that computes this maximum profit.

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

}

Last name(s) **Name** **ID**

(c) (1 pt.) We finally tackle the general problem, in which the sequence p can have any form and we want to compute the maximum profit $p_v - p_c$ that corresponds to buying the ticket on day c and selling it afterwards on day v. Explain at a high level how you would implement a function that computes this maximum profit and analyze its cost. Solutions with cost $\Omega(n^2)$ will be given 0 points.

Hint: the function of part b) can be useful to implement a divider-and-conquer algorithm.

This side would be intentionally blank if it were not for this note.