1. Cap 1: Introduction

1.1. Ejercicios 1.3.3.1

1.1.1. There are n webpages ($1 \leq n \leq 10M$). Page i has a page rank r_i. A new page ca be added or an existing page can be removed frequently. You want to pick the current top 10 pages with the highest page ranks, in order. Which method is better?

Load all n webpages page rank to memory, sort them in descending page rank ordern, and obtain the current top 10.

1.1.2. Given a list `L` with $100K$ integers, you need to frequently obtain `sum(i, j)`, i.e., the sum of `L[i] + L[i+1] + ... + L[j]`. Which data structure should you use?

Simple Array pre-processed with Dynamic Programming.

1.1.3. Given an $M \times N$ integer matrix $Q(1 \leq M, N \leq 70)$, determine if there exists a submatrix S of Q size $A \times B(1 \leq A \leq M, 1 \leq B \leq N)$ where $\texttt{mean}(S) = 7$. Which algorithm will not exceed $100M$ operations per test case in the worst case?

No sé.

1.1.4. Given a multiset S of $M = 100K$ integers, we want to know how many different integers that we can form if we pick two (no necessarily distinct) integers from S and sum them. The content of multiset S is prime numbers not more than $20K$. Which algorithm will not exceed $100M$ operations per test case in the worst case?

No sé

1.1.5. You have to compute the shortest path between two vertices on a weighted Directed Acyclic Graph (DAG) with $|V|, |E| \leq 100K$. Which algorithm(s) can be used?

- Breadth First Search.
- Dijkstra's

1.1.6. Which algorithm produces a list of the first $10M$ prime numbers with the best time complexity?

Sieve of Eratosthenes.

1.1.7. How to test if the factorial of n, i.e., $n!$ is divisible by an integer m ? $1 \leq n \leq 10^{14}$

- Test if $n! \% m == 0$

No funcionaría porque no se puede computar $10^{14}!$. Sin embargo, con mis conocimientos actuales, desconozco otra manera de hacerlo.

1.1.8. You want to enumerate all occurrences of a substring P (of length m) in a (long) string T (of length n), if any. Both n and m have a maximum of $1M$ characters. Which algorith is faster?

```
for (int i = 0; i < n-m; ++i){
    bool found = true;
    for (int j = 0; (j < m) && found; ++j  )
        if ((i+j >= n) || (P[j] != T[i+j]))
            found = false;
    if (found)
        printf("P is found at index %d in T\n", i);
}
```

1.1.9. Given a set S of N points scattered on a 2D plane ($2 \leq N \leq 5000$), find two points $\in S$ that have the greatest separating Euclidean distance. Is an $O(N^2)$ complete search algorithm that tries all possible pairs feasible?

Yes, such complete search is possible.

1.1.10. See Questin above, but now with a larger set of points: $2 \leq N \leq 200K$ and one additional constraint: The points are randomly scattered on a 2D plane

No sé

1.1.11. See the same Question above. We still have a set of $2 \leq N \leq 200K$ points. But this time there is no guarantee that the points are randomly scattered on a 2D plane.

No sé

1.2. Ejercicios 1.3.5.1

1.2.1. You receive a WA verdict for a very easy problem. What you should do?

- Create tricky test cases to find the bug.

Esto debido a que cuando el problema es muy fácil se tiende a omitir casos de prueba que no se toman en cuenta al momento de realizar la solución. La mayoría de veces estos casos especiales suelen ser un `if`, por lo tanto, son fáciles de sacar.

1.2.2.