1.a. i) $W_{IS} = \frac{N^2}{2} - \frac{N}{2}$

$A_{IS} = \frac{N^2}{4} + \sigma(N)$

ii) $N=100$

$inv(\sigma) + inv(\sigma^t) = \frac{N^2}{2} - \frac{N}{2}$

$inv(\sigma^t) = \frac{10000}{2} - 50 - 4500$

$= 450$

$inv(\sigma^t)^t = inv(\sigma) = 4500$

b) $T_1 \approx f \to \lim_{N\to\infty} \frac{T_1}{f} = 1$

$T_2 \approx f^2 \to \lim_{N\to\infty} \frac{T_2}{f^2} = 1$

i) $\lim_{N\to\infty} \frac{T_1^2}{T_2} = \lim_{N\to\infty} \frac{(T_1/f)^2}{T_2/f^2} = \frac{1}{1} = 1$ ✓

ii) $T_2 = f^2 + \sigma(T_1)$

False CX

$T_2 = N^4 + N^3$ $\quad N^3 \neq \sigma(N^2)$

$T_1 = N^2$

$f = N^2$

iii) $T_1 = \sigma(T_2)$

$\to \lim_{N\to\infty} \frac{T_1}{T_2}$

$= \lim_{N\to\infty} \frac{T_1/f^2}{T_2/f^2} = \frac{0}{1} = 0$

✓

c) $A^S_{2Search} = \sum_{i=1}^{N} \frac{1}{C_N} i^{-3/4} \cdot i = \frac{1}{C_N} \sum_{i=1}^{N} i^{1-3/4} = \frac{1}{C_N} \underbrace{\sum_{i=1}^{N} i^{1/4}}_{S_N}$

$\frac{1}{C_N} \sum_{i=1}^{N} i^{-3/4} = 1 \to C_N = \sum_{i=1}^{N} i^{-3/4}$

$\int_0^N x^{1/4} dx \leq S_N \leq \int_1^{N+1} x^{1/4} dx$

$\frac{4}{3} x^{5/4} \Big]_0^N \leq S_N \leq \frac{4}{5} x^{5/4} \Big]_1^{N+1}$

$\frac{4}{3} N^{5/4} \leq S_N \leq \frac{4}{5} (N+1)^{5/4} - \frac{4}{5}$

$S_N \approx \frac{4}{5} N^{5/4}$ because $\lim_{N\to\infty} \frac{S_N}{4/5 N^{5/4}} = 1$

$\int_0^N x^{-3/4} dx \geq C_N \geq \int_1^{N+1} x^{-3/4} dx$

$C_N \approx 4N^{1/4}$ $\quad A^S_{2Search} \approx \frac{\frac{4}{5} N^{5/4}}{4 N^{1/4}} = \frac{1}{5} N$

2 a) i) $\sum_{i=1}^{N} \log i \approx N \log N$ $\quad \sum_{i=1}^{N} \frac{1}{i} \approx \log N$

ii) N^2 in 1s with $N = 1000$ elements $\quad$ | 4s → 2000 elements

10000 elements $\quad 10^6 \to 1s$

$\underset{10}{k} \cdot 1000 \to k^2 = 100$ $\quad 10^8 \to 100s$

iii) $W_{Bsort} = \frac{N^2}{2} + \sigma(N)$

$A_{Bsort} = N-1$

b) $[\underbrace{k \ldots 1}_{①}, \underbrace{2k+1 \ldots 3k}_{②}, \underbrace{2k\ 2k-1\ \ldots, k+1}_{③}]$ $\quad N = 3k$

$n \geq inv(\sigma) = inv① + inv② + inv③ = k^2 - k + k^2 = 2k^2 - k = \frac{2N^2}{9} - \frac{N}{3}$

$inv① = \frac{k^2}{2} - \frac{k}{2}$

$inv② = inv②_{int} + inv②_{ext} = 0 + k \cdot k = k^2$

$inv③ = \frac{k^2}{2} - \frac{k}{2}$

$k=1 \quad 1, 3, 2 \quad inv = 1$ ✓

$k=2 \quad 2, 1, 5, 6, 4, 3 \quad inv = 6$

$n_{ot} \geq inv(\sigma^t)$

$inv(\sigma) + inv(\sigma^t) = \frac{N^2}{2} - \frac{N}{2} \to inv(\sigma^t) = \frac{N^2}{2} - \frac{N}{2} - \frac{2N^2}{9} + \frac{N}{3}$

$= \frac{9N^2 - 9N - 4N^2 + 6N}{18}$

$= \frac{5N^2 - 3N}{18}$

c) Min = $n-1$

max = $2n-2$

Algorithm Analysis, Computer Science EPS-UAM
Final Exam, First Part, January 2021

Last name: **Name:**
Group: **Room:** **Block:**

1	2	P. 1

Observations and warnings: read carefully before starting the exam

1. Write your name on **every page** of the exam and return **all of them** when you finish after carefully separating the pages that will be graded from those used as drafts. **Draft pages will not be graded in any case**. Proceeding otherwise will be considered as a possible cheating behavior.
2. You must **GUARD YOUR EXAM ACTIVELY**, keeping it out of the visual or physical reach of other students. Proceeding otherwise will be considered as a possible cheating behavior.
3. Cheating behavior detected during the exam or during the grading will be reported to the EPS academic directors so that they enforce the corresponding regulations.
4. **Students who did not take any mid-term exam**, and thus did not follow the continuous evaluation itinerary, must obtain an average grade of 7/10 in questions labeled as 1 in each part of the exam.
5. **Unless otherwise specified, only reasoned answers will be taken into account for the grading.**

Questions

1. a. (2 points) i. Write the worst and average cases of InsertSort.

   ii. If an array σ has 100 elements and 4,500 inversions, how many inversions does σ^t have? and $(\sigma^t)^t$?

   b. (3 points) If $T_1 \sim f$ and $T_2 \sim f^2$, both positive increasing functions tending to ∞ with N, reasonably determine whether the following statements are true or false:

   i. $\frac{T_1^2}{T_2} \sim 1$.
   ii. $T_2 = f^2 + \mathrm{O}(T_1)$.
   iii. $T_1 = \mathrm{o}(T_2)$.

   c. (5 points) The probability of searching the i–th element in a given array is

   $$P(K = T[i]) = \frac{1}{C_N} i^{-\frac{3}{4}},$$

   where C_N is a constant that guarantees that $\sum_1^N P(K = T[i]) = 1$.

   Reasonably estimate with the largest precision the asymptotic growth of the average case of successful linear searches $A^s_{LSearch}(N)$ on such array.

2. a. (3 points) i. What is the asymptotic equivalence of the sums $\sum_1^N \log n$ and $\sum_1^N \frac{1}{n}$ (**just enounce them**)?

   ii. An algorithm of cost n^2 takes one second to run on an array with 1,000 elements. How much time will it take to run on an array with 10,000 elements? And what would be the size of an array so that the algorithm takes 4 seconds to run on it?

   iii. What is the cost for the worst case of BubbleSort with swap control on N-element arrays? And the cost for the best case?

   b. (4 points) How many key comparisons would any local sorting algorithm perform at least on the following permutation?

   $\sigma = [K, ..., 1,\ \ 2K+1, ..., 3K-1, 3K,\ \ 2K, 2K-1, ...K+1]$ with $N = 3K$ elements?
   And on its **transpose** σ^t?

   c. (3 points) The following pseudocode corresponds to an algorithm that simultaneously calculates the maximum and the minimum of an array t with n elements:

```
int max_min(int t, int n):
    v_max = v_min = t[0]
    for i=1 to n-1:
        if t[i] > v_max:
            v_max = t[i]
        else if t[i] < v_min:
            v_min = t[i]

    return i_max, i_min
```

   Assuming key comparison as basic operation, reasonably calculate how many times will it run at most (maximum) and at least (minimum).

Algorithm Analysis, Computer Science EPS-UAM
Final Exam, Second Part, January 2021

Last name: **Name:**
Group: **Room:** **Block:**

1	2	P. 2

Questions

1. a. (3 points) i. What is the height of QuickSort's decision tree on 30-element arrays?

   ii. We would like to apply the Max Heap creation routine to an 8-element array. Indicate what is the maximum number of key comparisons that will be made during such creation.

   iii. What is the external path length of a complete binary tree with height 10?

   b. (3 points) After creating a max heap on a given permutation we obtain the following array:
   `[18 15 4 12 11 3]`.
   Argue that this array is indeed a max heap and sort it according to the second routine of the HeapSort algorithm by reasonably indicating each step.

   c. (4 points) Reasonably estimate the growth of a positive function T that meets $T(1) = 0$ and

   $$T(N) \leq \sqrt{N} + 9T\left(\left\lfloor \frac{N}{3} \right\rfloor\right).$$

   Estimate first a possible growth for an adequate special case and then use it to estimate the growth in the general case.

2. a. (2 points) i. What is the relationship between the average case $A_{XS}(N)$ of a key-comparison based sorting algorithm XS running on N-sized arrays and the external path length EPL of its decision tree T_{XS}^N?

   ii. With what permutation would we reach the worst case $W_{QS}(7)$ of QuickSort?

   b. (4 points) The following pseudocode corresponds to a version `msort_inv` of MergeSort where the **second subarray is sorted first** and the first subarray is sorted afterwards:

```
def msort_inv(array t, index f, index l):
    if f == l:
        return OK

    else:
        m = (f+l)/2             // integer division
        t_1 = msort_inv(t, m+1, l)
        t_2 = msort_inv(t, f, m)
        return combine(t_1, t_2, t)
```

   Write the decision subtree for 4-element arrays when the algorithm is applied to arrays σ in which $\sigma(2) = 4$.

   c. (4 points) The following recursive pseudocode calculates the value of the n-th Fibonacci number:

```
int fib(int t):
    if n == 0 or n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)
```

   We would like to reasonably estimate how many **sums** as a function of n this algorithm performs to calculate the n-th Fibonacci number. For this goal, first provide the number of sums for $n = 0, 1, 2, 3$ and 4, and then estimate the value of the general sum number.

Algorithm Analysis, Computer Science EPS-UAM
Final Exam, Third Part, January 2021

Last name: **Name:**
Group: **Room:** **Block:**

P. 1	P. 2	1	2	P. 3	T

Questions

1. a. (3 points) i. What are the worst and average cases of the binary search algorithm on N-sized arrays?

   ii. Although the cost of a binary search on sorted arrays is optimal, it is not a good idea to use such arrays as a data structure for a Dictionary ADT. Why?

   iii. Briefly indicate the steps to remove a node with two children from a binary search tree.

   b. (4 points) Reasonably build the AVL associated to the array:
   `[20 21 13 10 11 6 9]`
   indicating with sufficient detail at each step the unbalance to be fixed and the corresponding actions to do it.

   c. (3 points) We would like to build a hash table with chaining to store a given number of data N by using an array of 300 pointers. What is the maximum value of N so that the average number of probes in successful and unsuccessful searches is at most 1.5? And what would be that maximum value if we consider a hash table with open addressing and random probing?

2. a. (3 points) i. Of what order is the height of an AVL with N nodes?

   ii. How would you define a **multiplication** hash function so that for an integer M provides values between 0 and $M - 1$?

   iii. For a given hash with chaining H it is met that $A_H^f(N, m) = \phi(\lambda)$ for a given function ϕ and $\lambda = N/m$, where N is the number of data to be stored and m is the number of pointers. How could we express $A_H^s(N, m)$ as a function of ϕ?

   b. (3 points) We would like to use an 11-element array, the hash function $h(k) = k\%11$, indexes between 0 and 10, and chaining to store the keys: `[17, 11, 22, 6, 3, 14]`. Indicate the final state of the array and the number of collisions that occur during its construction.

   c. (4 points) From a given hash algorithm H with **open addressing** we know that

   $$A_H^f(N, m) = \frac{1}{(1-\lambda)^2},$$

   with $\lambda = N/m$ being the load factor.

   Reasonably deduce what would be the formula for the average number $A_H^s(N, m)$ of probes needed to successfully search in an array with N data and m positions. What is the average number of probes in successful searches for arrays with a load factor $\lambda = 0.5$?