CSE202 – FINAL EXAM

The 4 exercises are independent and can be treated in arbitrary order. Within an exercise, the answer to a question can be used in the next ones even if a proof has not been found.

The number of points indicated in front of each question is an indication of the relative difficulties of the questions. It is not necessary to solve all the questions to obtain the maximal possible grade.

Exercise 1. *Let $C(N)$ be the best case for the number of comparisons in an unsuccessful binary seach in a table of length N.*

(1) *(1 pt) Show that $C(N) = 1 + C(\lfloor (N-1)/2 \rfloor)$;*
(2) *(1 pt) Give a formula for largest N such that $C(N) = k$.*

Solution : 1. In binary search, one first picks an element in the middle and performs one comparison with that element. When the search is unsuccessful, recursion takes place on one of the "halves" of the array, whose exact sizes are $\lfloor (N-1)/2 \rfloor$ and $\lceil (N-1)/2 \rceil$. The best case is when at each stage, the recursion takes place in the smaller of the two halves. The complexity therefore satisfies the recurrence

$$C(N) = 1 + C\left(\left\lfloor \frac{N-1}{2} \right\rfloor\right), \quad C(0) = 0, C(1) = 1.$$

2. This recurrence gives the first values of C as $0, 1, 1, 2, 2, 2, 2, 3$, leading to $N_1 = 2, N_2 = 6$. More generally, a maximal sequence N_k for which $C(N_k) = k$ is thus obtained when at each stage the floor is smaller than the value, leading to

$$N_0 = 0, \qquad N_{k+1} = 2(N_k + 1).$$

The solution to this recurrence is $N_k = 2^{k+1} - 2$. □

Exercise 2. *Consider a trie built from n random keys over the alphabet $\{0, 1\}$.*

(1) *(2 pts) Show that the average number of nodes examined in an unsuccessful search for an infinite binary string is*

$$\sum_{k \geq 0} (1 - (1 - 2^{-k})^n).$$

(2) *(2 pts) Prove that this number is upper bounded by*

$$\log_2 n + 4.$$

Hint: split the sum into the part for k from 0 to $\lfloor \log_2 n \rfloor$ and its tail, and use the simple bounds

$$e^{-u} \geq 1 - u \quad \text{for } u \in \mathbb{R}, \qquad \log(1 - x) \geq -3x/2 \quad \text{for } x \in [0, 1/2].$$

Solution : 1. The probability that two bit strings differ in one of their first k bits is $1-2^{-k}$. Thus the probability that all of the n keys in the trie differ with the searched key in one of the first k bits is $(1-2^{-k})^n$, which is therefore $\mathbb{P}(\text{cost} \leq k) = 1 - \mathbb{P}(\text{cost} > k)$. Thus the average number of nodes examined in an unsuccessful search is

$$\begin{aligned}\sum_{k\geq 1} k\,\mathbb{P}(\text{cost}=k) &= \sum_{k\geq 1} k\,(\mathbb{P}(\text{cost}>k-1) - \mathbb{P}(\text{cost}>k)) \\ &= \sum_{k\geq 0} \mathbb{P}(\text{cost}>k) = \sum_{k\geq 0}(1-(1-2^{-k})^n),\end{aligned}$$

where the reorganisation of terms in the infinite sum is justified by the fast decrease of $\mathbb{P}(\text{cost} > k)$.

2. The summands $(1-(1-2^{-k})^n)$ are all upper bounded by 1, so that

$$\sum_{k=0}^{\lfloor \log_2 n \rfloor} (1-(1-2^{-k})^n) \leq \log_2 n + 1.$$

For the tail, the bound $\log(1-x) \geq 3x/2$, multiplied by n and exponentiated, gives

$$1-(1-x)^n \leq 1 - \exp(-3nx/2) \leq 3nx/2$$

so that

$$\sum_{k>\log_2 n} (1-(1-2^{-k})^n) \leq \frac{3}{2} n \sum_{k>\log_2 n} 2^{-k} \leq \frac{3}{2} n 2^{-\log_2 n} \underbrace{\sum_{k\geq 0} 2^{-k}}_{2} = 3.$$

□

Exercise 3. *Keeping an exact count of a huge number N of events (e.g., packets of data passing through a router) uses $\log N$ bits of memory. Consider the following probabilistic algorithm computing an approximation of N using only $\log\log N$ bits of memory: initialize $k=0$ and at each new event, increment k with probability 2^{-k}. At the end of the algorithm, return $2^k - 1$. The aim of this exercise is to estimate how good such an algorithm can be.*

Let C_n be the random variable representing the content of the counter after n events and $p_{n,\ell}$ be the probability that $C_n = \ell$.

(1) *(1 pt) Show that*

$$p_{n+1,\ell} = (1-2^{-\ell})p_{n,\ell} + 2^{-(\ell-1)} p_{n,\ell-1}.$$

(2) *(3 pts) Deduce by induction that the expectation of 2^{C_n} is $n+1$, the expectation of 2^{2C_n} is $3n(n+1)/2+1$ and the variance of 2^{C_n} is $n(n-1)/2$.*

In order to improve the precision by reducing the variance, consider taking t counters for which the same algorithm is applied in parallel. At the end, the average value $Z_n^{(t)}$ of the estimates $2^{k_1}, \dots, 2^{k_t}$ is returned.

(3) *(2 pts) Show that there exists a value of t, independent of n, such that*

$$\mathbb{P}\left(\left| \frac{Z_n^{(t)}}{n+1} - 1 \right| \geq \frac{1}{4} \right) \leq \frac{1}{4}.$$

In other words, in more than 75% of the executions, the result is within 25% of the exact value.
[Hint: Recall that Chebyshev's inequality states that if a random variable X has finite expectation m and finite non-zero variance σ^2, then for any real $k > 0$, $\mathbb{P}(|X - m| \geq k\sigma) \leq 1/k^2$.]

Solution : 1. The recurrence expresses that the counter is equal to ℓ after $n+1$ events either when it was equal to ℓ after n events (which occurs with probability $p_{n,\ell}$) and stayed at this value (with probability $1 - 2^{-\ell}$) or when it was equal to $\ell - 1$ (probability $p_{n,\ell-1}$) and increased its value (probability $2^{-(\ell-1)}$).

2. For $n = 0$, $C_0 = 0$ and thus $2^{C_0} = 1$ deterministically. If the formula holds up to n, then

$$\begin{aligned}\mathbb{E}(2^{C_{n+1}}) &= \sum_{\ell \geq 0} 2^\ell p_{n+1,\ell} \\ &= \sum_{\ell \geq 0} 2^\ell (1 - 2^{-\ell}) p_{n,\ell} + \sum_{\ell \geq 0} 2^\ell 2^{-(\ell-1)} p_{n,\ell-1} \\ &= \mathbb{E}(2^{C_n}) - 1 + 2 = n + 2,\end{aligned}$$

which concludes the proof for the expectation. For the variance, one can first compute

$$\begin{aligned}\mathbb{E}((2^{C_{n+1}})^2) &= \sum_{\ell \geq 0} 2^{2\ell} p_{n+1,\ell} = \sum_{\ell \geq 0} 2^{2\ell} (1 - 2^{-\ell}) p_{n,\ell} + \sum_{\ell \geq 0} 2^{2\ell} 2^{-(\ell-1)} p_{n,\ell-1} \\ &= \mathbb{E}((2^{C_n})^2) - \mathbb{E}(2^{C_n}) + 4\mathbb{E}(2^{C_n}) = \mathbb{E}((2^{C_n})^2) + 3(n+1),\end{aligned}$$

which gives a recurrence for $\mathbb{E}((2^{C_n})^2)$ from which it follows that

$$\mathbb{E}((2^{C_n})^2) = \frac{3}{2} n(n+1) + 1$$

and therefore the variance is

$$\mathbb{E}((2^{C_n})^2) - (\mathbb{E}(2^{C_n}))^2 = \frac{n(n-1)}{2}.$$

3. By linearity, the expectation of $Z_n^{(t)}$ is $n+1$. Since the counters are independent random variables, its variance is $n(n-1)/(2t) \leq (n+1)^2/(2t)$. Thus by Chebyshev's inequality, for any $k > 0$,

$$\begin{aligned}\mathbb{P}\left(\left|\frac{Z_n^{(t)}}{n+1} - 1\right| \geq \frac{k}{\sqrt{2t}}\right) &= \mathbb{P}\left(|Z_n^{(t)} - n - 1| \geq \frac{k(n+1)}{\sqrt{2t}}\right) \\ &\leq \mathbb{P}\left(|Z_n^{(t)} - n - 1| \geq \frac{k\sqrt{n(n-1)}}{\sqrt{2t}}\right) \leq \frac{1}{k^2}.\end{aligned}$$

Therefore the choice $t = (4)^3/2 = 32$ is sufficient. □

Exercise 4. *The Knapsack Problem is the problem of packing the maximal value within a bounded weight limit. Formally, it is stated as follows:*

Given positive integers $W, w_1, \ldots, w_\ell$ and $v_1, \ldots, v_\ell$, find a subset $S \subset \{1, \ldots, \ell\}$ that maximizes $\sum_{i \in S} v_i$ subject to $\sum_{i \in S} w_i \leq W$.

Without loss of generality, we assume that $W \geq \max(w_1, \ldots, w_\ell)$.

(1) *(1 pts) Show that the Knapsack Problem is NP-hard.*
(2) *(4 pts) Let $S_{j,v}$ be the smallest weight over subsets of $\{1,\dots,j\}$ whose total value is v. Show that $S_{j,v} = \min(S_{j-1,v-v_j} + w_j, S_{j-1,v})$. Deduce from there an algorithm that solves the problem in time $O(\ell^2 v_{\max})$, where $v_{\max} = \max(v_1,\dots,v_\ell)$. Explain why this complexity does not contradict the NP-hardness.*
(3) *(3 pts) Exploit this idea to design a FPTAS for the problem, using values $(\lfloor v_1/K\rfloor,\dots,\lfloor v_\ell/K\rfloor)$ for a scaling factor K to be determined.*

Solution : 1. It is sufficient to find a reduction from the SubsetSum Problem. Given $(x_1,\dots,x_\ell) \in \mathbb{N}^\ell$ and $k \in \mathbb{N}$, take $w_i = v_i = x_i$ for all i and $W = k$.

2. In the subsets of $\{1,\dots,j\}$ reaching the total value v with minimal weight, either the jth element is not used and the subset is the one that was found with subsets of $\{1,\dots,j-1\}$ and its weight is $S_{j-1,v}$, or the jth element is added to a subset of $\{1,\dots,j-1\}$ with total value $v - v_j$ and minimal weight, whence the equation. The resulting dynamic programming algorithm is as follows:

```
def knapsack(w,v,W):
    ell = len(w)
    vmax = max(v)+1
    S = [[]]*(ell*vmax)
    Weight = [0]+[math.inf]*(ell*vmax-1)
    for j in range(ell):
        NewWeight = Weight.copy()
        newS = S.copy()
        for vv in range(ell*vmax-v[j]):
            if Weight[vv]+w[j]<NewWeight[vv+v[j]]:
                newS[vv+v[j]]=S[vv]+[j]
                NewWeight[vv+v[j]]=Weight[vv]+w[j]
        Weight = NewWeight
        S = newS
    for vv in range(ell*vmax-1,-1,-1):
        if Weight[vv]<=W: return vv,S[vv]
    return None
```

The correctness of this algorithm comes from the fact that the sum of values is bounded by $\ell v_{\max}$.

The complexity is as announced: there is one loop running from 0 to ℓ within which another one performs $O(\ell v_{\max})$ steps. This complexity is not polynomial, since $v_{\max}$ is potentially exponential in the bit-size of the input.

3. If one first divides all values by a fixed K and runs the algorithm with the values $(v'_1,\dots,v'_\ell) = (\lfloor v_1/K\rfloor,\dots,\lfloor v_\ell\rfloor)$, then the algorithm runs in time $O(\ell^2 v_{\max}/K)$.

Since all the values v'_i are such that $Kv'_i \le v_i$, the optimal value found by running on the v'_i is smaller than the optimal one. Moreover, for any i, by the usual properties of the floor function, $Kv'_i > v_i - K$. Therefore, if Opt' and Opt denote the optimal sets for the approximate and for the exact algorithm, the following inequalities are satisfied:

$$\sum_{i\in\mathrm{Opt}'} v_i \ge K \sum_{i\in\mathrm{Opt}} v'_i \ge \sum_{i\in\mathrm{Opt}} v_i - \ell K \ge V_{\mathrm{opt}} - \ell K,$$

where V_{opt} is the optimal value.

For any $\epsilon > 0$, choosing $K = \epsilon v_{\max}/\ell \leq \epsilon V_{\text{opt}}/\ell$ ensures that

$$V_{\text{opt}} - \ell K \geq V_{\text{opt}}(1 - \epsilon),$$

and the complexity is $O(\ell^3/\epsilon)$, yielding a FPTAS for this problem. □