Homework 1

Lina Brilliantova, restel

Semptember 2020

Problem 1

Part A

The total number of anagrams, $c(A)$, is the number of unique permutations of n characters. If a single letter is repeated in the word k times, then there exists $k!$ ways to permute the repeated letters and get the identical anagrams. If k letters are repeated, k_i times each, than the total number of non-distinct anagrams is a multiplication of factorials of the corresponding k_i.

$$c(A) = \frac{n!}{k_1! \times k_2! \times ...k_k!} \tag{1}$$

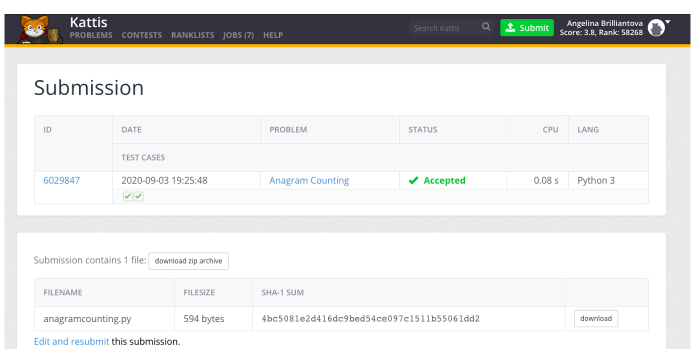
where k_i is the number of characters of type i, $i \in [0, k]$, k is the number of different letters.

So, the algorithm iterates over n characters and count the number of each letter and then calculates $c(A)$ according to Equation 1.

Running time

$\mathcal{O}(n)$ as the algorithm iterates over all given characters.

Printscreen



1 Problem 2

Let CH = [] be the Convex Hull of the given set of n points;
Let *alloc* be the set of unique fair allocations.;
for *each point* $pt \in CH$ **do**
    ; // $\mathcal{O}(n)$
    draw a vertical half-line up, l_{pt} ;
    for $\forall i \in [1, n]$ **do**
        draw a line from pt to point i, l_i ;
        compute an angle, δ, between l_{pt} and l_i ;
    Let A be an array of n points sorted radially by δ in clock-wise order. ; // $\mathcal{O}(n \log n)$
    Find two median points in A ;
    Let L_i be $A[1, n/2 - 1]$;
    Let S_i be $A[n/2, n]$;
    if L, S *are contained in alloc* **then**
        $alloc.append((L, S))$

Running time

$\mathcal{O}(n^2 \log n)$

Perfomance

I think, I have an implementation issue, as I could not listed the points in a clockwise order. If I manually check, the algorithm works correctly on all smaller inputs with $n < 8$.

Input	Correct answer	My output
1	6	6
2	2	2
3	2	4
4	4	4
5	6	6
6	12	10
7	76	100
8	118	100
9	1346	1000
10	2	2000
11	2610	2000
12	2	1000

Table 1: Caption

Total correct answers: 4

Output of Kattis: Wrong Answer

2 Problem 3

Part A) Counting

WHAT

$S[v,k]$ - the number of subsets chosen from k first items, that include k item, and where these items have total weight $\leq v$

HOW

$$S[v,k] = \begin{cases} 0 & \text{if } k = 0 \vee v = 0 \vee w_k > v. \\ \sum_{i=0}^{k-1} S[v-w_k, i] + 1 & \text{otherwise} \end{cases} \tag{2}$$

When considering $S[v,k]$ with $k \neq 0$, $v \neq 0$, $w_k > v$, any subset of items chosen from $< k$ items, that fit into $v - w_k$ can accommodate k item, so to get the total number of subsets we are summing up all entries of the table with row $v - w_k$ and column less than k. We are adding up 1 to account for the subset, consisting of just k item.

WHERE

$$\sum_{i=1}^{n} S[W,i] + 1 \tag{3}$$

$S[W,i]$ - is the number of non-overlapping subsets fitting into W. If we vary i from 1 to the number of items, n, then we will get the total number of subsets. We add up 1 to account for the empty set.

Running time

$\mathcal{O}(n^2 W)$

S has to be computed for each combination of $v \in [0,W]$ and $k \in [0,n]$. The computation of each $S[v,k]$ takes $\mathcal{O}(n)$, as the entries have to be sum up to k.

Part B) Sampling

Sample each item i proportionally to its probability, calculated from the dynamic programming table.

Input : Array S with $W+1$ rows and $n+1$ columns. Each element $S[v,k]$ represent the number of subsets chosen from k first items, including k item, with total weight $\leq v$ [WHAT, part A]

Output: A subset of items with total weight $\leq$ W chosen uniformly at random

```
i = n ;
Z = W ;
subset = [] ;
while i > 0 do
    total = Σ_{j=1}^{i} S[Z, j] + 1 ;                          // O(n)
    Let c be a random number in [0, 1) ;
    if c < S[Z,i]/total then
        subset.append(i) ;                                  // choose item i
        Z = Z − w_i ;                       // update the total allowed weight
    i = i − 1 ;                                    // go to the previous item
```

Running time

$\mathcal{O}(n^2)$ counting total is $\mathcal{O}(n)$ operation. We count total for $i \in [0, n]$.

Perfomance

Part A) Counting

Input	Success	Time (sec)
1	+	0.0015
2	+	0.0037
3	+	0.37
4	+	10.0
5	+	9.65

Part B

The probability that a random subset includes the n-th item, $P(n)$, can be computed as:

$$P(n) = \frac{S[W,n]}{\sum_{j=1}^{n} S[W,j])}$$

R_k converges to $P(n) = 0.429$ as n increases. If the weight of $n-$th item is replaced by 1, $P(n)$ changes to 0.5 and less samples are needed for convergence, as now it is a more frequent event, that can be on average observed on smaller samples.

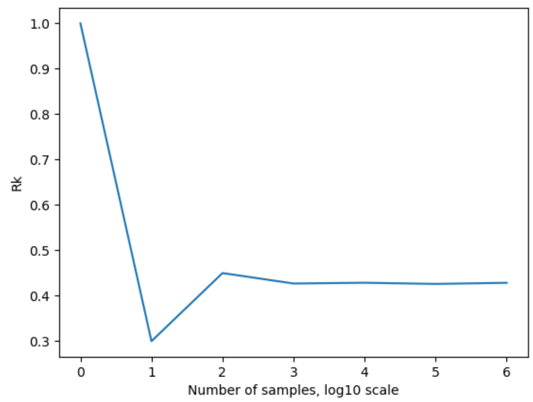


Figure 1: The convergence of R_k for unmodified input 2. The values are $[1., 0.3, 0.45, 0.427, 0.4287, 0.42612, 0.428579]$

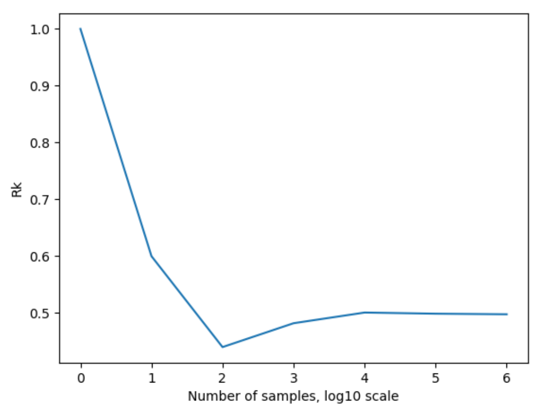


Figure 2: The convergence of R_k for modified input 2 with $w_n = 1$. The values are $[1., 0.6, 0.44, 0.482, 0.5008, 0.49879, 0.497766]$