CS 473 ✧ Fall 2022

Homework 1

Due Tuesday, September 6, 2022 at 9pm

- Starting with this homework, groups of up to three students can submit joint solutions for each problem. For each numbered problem, exactly one member of each homework group should submit a solution and identify the other group members (if any) on Gradescope. Please also list all group members at the top of the first page of every submission.

---

1. Suppose we are given a bit string $B[1\,..\,n]$. A triple of indices $1 \le i < j < k \le n$ is called a ***well-spaced triple*** if $B[i] = B[j] = B[k] = 1$ and $k - j = j - i$.

   (a) Describe a brute-force algorithm to determine whether B has a well-spaced triple in $O(n^2)$ time.

   (b) Describe an algorithm to determine whether B has a well-spaced triple in $O(n \log n)$ time. *[Hint: FFT!]*

   (c) Describe an algorithm to determine the *number* of well-spaced triples in B in $O(n \log n)$ time.

2. This problem explores different algorithms for computing the factorial function $n! = 1 \cdot 2 \cdot 3 \cdots n$. This may seem like a weird question; the obvious algorithm uses n multiplications, and thus runs in $O(n)$ time. Right?

   Well, actually, no. The inequalities $(n/2)^{n/2} < n! < n^n$ imply that the exact binary representation of $n!$ has length $\Theta(n \log n)$. So the *number* of multiplications is not a good estimate of the actual time required to compute $n!$; we also need to account for the *time* for those multiplications.

   (a) Recall that the standard lattice algorithm that you learned in elementary school multiplies any n-bit integer and any m-bit integer in $O(mn)$ time. Describe and analyze a variant of Karatsuba's algorithm that multiplies any n-bit integer and any m-bit integer, for any $n \ge m$, in $O(n \cdot m^{\lg 3 - 1}) = O(n \cdot m^{0.58496})$ time.

   (b) Consider the following classical algorithm for computing the factorial $n!$ of a non-negative integer n:

```
FACTORIAL(n):
  fact ← 1
  for i ← 1 to n
        fact ← fact · i        (∗)
  return fact
```

   Analyze the running time of FACTORIAL(n) using different algorithms for the multiplication in line $(*)$:

   i. Lattice multiplication
   ii. Your variant of Karatsuba's algorithm from part (a)

(c) The following divide-and-conquer algorithm also computes the factorial function, but using a different grouping of the multiplications. The subroutine FALLING computes the *falling power* function $n^{\underline{m}} = n(n-1)(n-2)\cdots(n-m+1) = n!/(n-m)! = \binom{n}{m}\cdot m!$:

```
FALLING($n, m$):
  if $m = 0$
      return 1
  else if $m = 1$
      return $n$
  else
      return FALLING($n, \lfloor m/2 \rfloor$) $\cdot$ FALLING($n - \lfloor m/2 \rfloor, \lceil m/2 \rceil$)
```

```
FASTERFACTORIAL($n$):
  return FALLING($n, n$)
```

Analyze the running time of FASTERFACTORIAL(n) using different algorithms for the multiplication in the last line of FALLING:

i. Lattice multiplication

ii. Your variant of Karatsuba's algorithm from part (a)

(For simplicity, assume n is a power of 2 and ignore the floors and ceilings.)

3. Your new boss at the Dixon Ticonderoga Pencil Factory asks you to design an algorithm to solve the following problem. Suppose you are given N pencils, each with one of c different colors, and a non-negative integer k. ***How many different ways are there to choose a set of k pencils?*** Two pencil sets are considered identical if they contain the same number of pencils of each color.

For example, suppose you have two red pencils, four green pencils, and one blue pencil. Then you can form exactly five different two-pencil sets (RR, RG, RB, GG, GB), exactly six different four-pencil sets (RRGG, RRGB, RGGG, RGGB, GGGG, GGGB), and exactly three different six-pencil sets (RRGGGG, RRGGGB, RGGGGB).

Describe an algorithm to solve this problem, and analyze its running time. Your input is an array $Pencils[1..c]$ and an integer k, where $Pencils[i]$ stores the number of pencils with color i. Your output is a single non-negative integer. For example, given the input $Pencils = [2,4,1]$ and $k = 2$, your algorithm should return the integer 5.

For full credit, report the running time of your algorithm as a function of the parameters N (the total number of pencils), c (the number of colors), and k (the size of the target pencil sets). Assume that $k \ll c \ll N$, but do not assume that any of these parameters is a constant. *Assume for this problem that all arithmetic operations take $O(1)$ time.*

Hint:

$$\underbrace{(1+x+x^2)}_{\text{2 red pencils}}\cdot\underbrace{(1+x+x^2+x^3+x^4)}_{\text{4 green pencils}}\cdot\underbrace{(1+x)}_{\text{1 blue pencil}} = 1+3x+\underbrace{5x^2}_{\text{5 2-pencil sets}}+6x^3+\underbrace{6x^4}_{\text{6 4-pencil sets}}+5x^5+\underbrace{3x^6}_{\text{3 6-pencil sets}}+1$$