Homework #2

Due Time: 2015/10/30 (Fri.) 12:00
Contact TAs: ada@csie.ntu.edu.tw

Instructions and Announcements

- For the first time submitting your ADA 2015 HW, please create a repository for ADA on bitbucket (https://bitbucket.org) and share the repository with user ada2015 with read permission.
- You have to login to the judge system (http://ada2015.csie.org) to bind the bitbucket repository to your account. For programming problems (those containing "**Programming**" in the problem topic), push your source code to your bitbucket repository. After you push the source code to repository, you can run the judge and get the score. Please refer to the guide of the judge system on its index page for further details.
- For other problems (also known as "hand-written problems"), submit the answers in an electronic copy via the git repository before the due time. Please combine the solutions of all these problems into **only ONE file** in the PDF format, with the file name in the format of "hw[# of HW].pdf" (e.g. "hw1.pdf"), all in **lowercase**; otherwise, you might only get the score of one of the files (the one that the grading TA chooses) or receive penalty because of incorrect filename.
- Discussions with others are strongly encouraged. However, you should write down your solutions in your own words. In addition, for **each and every** problem you have to specify the references (the Internet URL you consulted with or the people you discussed with) on the first page of your solution to that problem. You may get zero point for problems with no specified references.
- **NO LATE SUBMISSION ONE DAY AFTER THE DUE TIME IS ALLOWED.** For all submissions, up to one day of delay is allowed; however, penalty will be applied to the score according to the following rule (the time will be in seconds):

$$\text{LATE\_SCORE} = \text{ORIGINAL\_SCORE} \times (1 - \text{DELAY\_TIME}/86400)$$

Note that late submission of partial homework is NOT allowed. The penalty will be applied to the entire homework in the case of late submission.

Problem 1

Dynamic programming is also a good choice for combinatorial problems. For these types of problems, it's helpful to write down recurrence relations first, and then the algorithm is usually straightforward.

When analysing time complexity, **assume every basic arithmetic operation (i.e. addition, subtraction, multiplication, and division) takes $O(1)$ time**.

(1) **Binomial Numbers** (5%)

Let $\binom{n}{m}$ be the number of ways to choose m things among n different things.

For example: $\binom{0}{0} = 1, \binom{2}{1} = 1, \binom{4}{2} = 6$.

Design an algorithm to calculate $\binom{n}{m}$ for all $0 \le m \le n \le N$ in $O(N^2)$ time.

(2) **Catalan Numbers** (7%)

Let C_n be the number of ways to go from $(0, 0)$ to (n, n) on the (x, y)-plane, with the following constraints:

- You can only go 1 unit right ($x \to x + 1$) or upward ($y \to y + 1$) every step.
- You must not touch any point (x, y) with $x < y$.

The first few terms of C_n are: $1, 2, 5, 14, 42, \cdots$.

Design an algorithm to calculate C_n for all $1 \le n \le N$ in $O(N^2)$ time.

Hint: Let $f(n, m)$ be the number of different ways to go from $(0, 0)$ to (n, m). Can you derive the recurrence relation?

(3) **Partition Numbers** (8%)

Let P_n be the number of ways to partition n **identical** things into groups.

Formally, P_n is the number of different sequences $\{a_1, a_2, \cdots, a_k\}$ such that $k \ge 1$, $a_1 \ge a_2 \ge \cdots \ge a_k \ge 1$ and $a_1 + a_2 + \cdots + a_k = n$.

For example, $P_5 = 7$, because $5 = 4{+}1 = 3{+}2 = 3{+}1{+}1 = 2{+}2{+}1 = 2{+}1{+}1{+}1 = 1{+}1{+}1{+}1{+}1$.

The first few terms of P_n are: $1, 2, 3, 5, 7, 11, 15, \cdots$.

Design an algorithm to calculate P_n for all $1 \le n \le N$ in $O(N^3)$ time.

Hint 1: Let $f(n, m)$ be the number of ways to partition n into groups of size at most m. (That is, all $a_i \le m$ in the above definition.)

Hint 2: Among all partitions in $f(n, m)$, how many of them have exactly k groups of size m? Express the number in terms of $f(\cdot, \cdot)$.

Problem 2

HH-code is a kind of string with `0` and `1`. For a given HH-code H, any subsequence of H is called a hidden HH-code of H.

For example:

If there is an HH-code H = `1011`, then `1, 0, 10, 01, 11, 101, 111, 011, 1011` are all the **different** hidden HH-codes of H.

When analysing time complexity, **assume every basic arithmetic operation (i.e. addition, subtraction, multiplication, and division) takes $O(1)$ time**.

(1) **Hidden HH-Code** (5%)
If the length of HH-Code is N, please design an algorithm to calculate the number of different hidden HH-Codes in $O(N^2)$ time.

(2) **Hidden HH-Code Fast** (8%)
Please design an algorithm to calculate the number of different Hidden HH-codes in $O(N)$ time.
Hint: Speed up the last problem with partial sum, or you could do it in O(N) time directly.

(3) **Hidden HH-Code with length** (7%)
The length of HH-code is still N, but we want to know the number of different hidden HH-code with a specific length K. Please design an algorithm to calculate the number in $O(KN)$ time.

Problem 3

In this problem, we will learn some tricks to optimize dynamic programming algorithms.

(1) Fast Matrix Exponentiation (10%): Sometimes, the recursive formula in a dynamic programming problem is a linear combination of the answers in the previous state. That is, the recursive formula has the form

$$\begin{bmatrix} \texttt{dp}_1(n) \\ \texttt{dp}_2(n) \\ \vdots \\ \texttt{dp}_m(n) \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,m} \end{bmatrix} \begin{bmatrix} \texttt{dp}_1(n-1) \\ \texttt{dp}_2(n-1) \\ \vdots \\ \texttt{dp}_m(n-1) \end{bmatrix}$$

For example, let $\texttt{dp}_1(n) = f(n)$ and $\texttt{dp}_2(n) = f(n-1)$, the formula below calculate the n-th fibonacci number with $f(0) = 0, f(1) = 1$.

$$\begin{bmatrix} f(n) \\ f(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} f(n-1) \\ f(n-2) \end{bmatrix}$$

When analysing the time complexity in this problem, assume that each of the arithmetic operations $+, -, \times, /$ takes $\mathcal{O}(1)$ time.

1. (4%) Prove that for an $m \times m$ matrix A, both A^n and $\sum_{i=0}^{n} A^i$ could be calculated in $\mathcal{O}\left(m^3 \log_2 n\right)$ time.

   Hint: You should have learned how to calculate A^n from homework 1.
   To calculate $\sum_{i=0}^{n} A^i$, consider the matrix $\begin{bmatrix} A & I \\ 0 & I \end{bmatrix}$.

2. (6%) Solve the following problem.

   Recently Saki has discovered some new creatures that come from a new world. She analyses their DNA sequence and has the following findings.

   (a) Unlike the livings on earth which have only 4 different kinds of nucleotides, they have K different kinds of nucleotides, $\Gamma = \{n_1, n_2, \ldots, n_K\}$.
   (b) Some nucleotide pairs cannot appear consecutively due to certain chemical effects. In other words, there are Q pairs of $(\beta_{i,1}, \beta_{i,2})$ $(1 \le i \le Q,\ \beta_{i,\{1,2\}} \in \Gamma)$ such that $\beta_{i,1}\beta_{i,2}$ and $\beta_{i,2}\beta_{i,1}$ will never appear in the DNA sequence.
   (c) In each DNA sequence, there are P positions that are immutable: The nucleotide at position a_j $(1 \le j \le P)$ in the sequence will always be $b_j \in \Gamma$.
   (d) The length of the DNA sequence λ satisifies $\max(a_i) \le L \le \lambda \le R$ where L and R are two constant numbers.

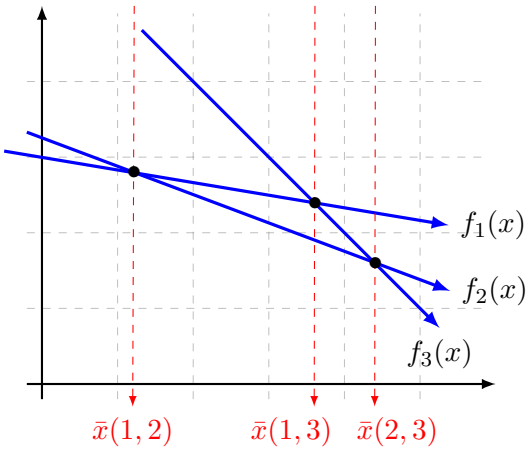
   Saki wonders how many possible DNA sequences exist. Please design an algorithm to solve the problem in $\mathcal{O}\left(K^3 P \log_2 R\right)$ time.
   Hint: Try to formulate your problem using the above matrix representation.

(2) Convex Hull Optimization (20%): For dynamic programming problems of the form

$$\texttt{dp}(i) = \min_{j<i} a_j x_i + b_j, \tag{1}$$

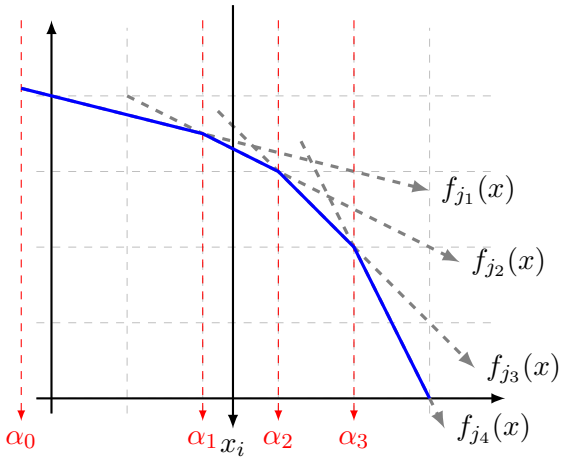
where $\langle a_i \rangle, \langle b_i \rangle, \langle x_i \rangle$ are three sequences with a_i **decreasing** and x_i **increasing**. It is easy to calculate all the `dp` values $\texttt{dp}(1), \texttt{dp}(2), \cdots, \texttt{dp}(n)$ in $\mathcal{O}(n^2)$ time, but we could do it better.

1. (3%) Let $f_j(x) = a_j x + b_j$, so that $\texttt{dp}(i) = \min_{j<i} f_j(x_i)$. Define $\bar{x}(j,k)$ for all $j < k$ to be the minimal x such that $f_k(x) \le f_j(x)$.
   That is, if $x_i \ge \bar{x}(j,k)$, then for $\texttt{dp}(i)$, $f_k(x_i) \le f_j(x_i)$, so it would be better to "transfer" from k rather than j. Notice that since a_j is decreasing, $\bar{x}(j,k)$ always exist.
   Please give the equation of $\bar{x}(j,k)$.

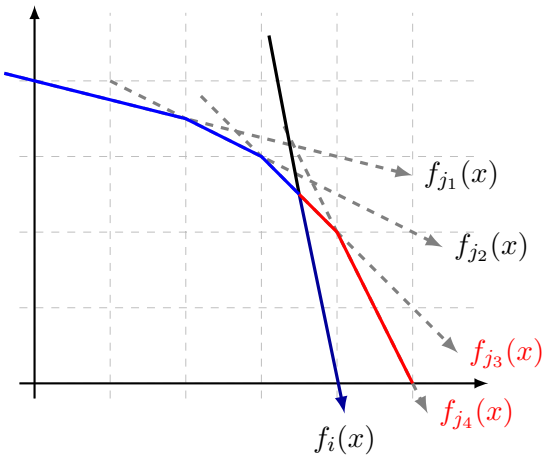


2. (10%) Describe an algorithm to calculate $\texttt{dp}(i)$ for all $1 \le i \le n$ in $\mathcal{O}(n)$ time.

Hint: See $f_j(x)$ as a line. Use a double-ended queue (deque) to maintain a sequence $(j_1, j_2, \cdots, j_m)$, such that $j_1 < j_2 < \cdots < j_m$, and there exists $\alpha_0 < \alpha_1 < \cdots < \alpha_m = \infty$ such that $\alpha_0 \le x_k$ and $f_{j_\gamma}(x)$ is the minimum of $f_1(x), f_2(x), \cdots, f_{k-1}(x)$ for $\alpha_{\gamma-1} \le x \le \alpha_\gamma$. If this could be achieved, how to calculate $\texttt{dp}(i)$?
Finally, how to add a new line (namely, $f_i(x)$) into the deque?



Calculating $\texttt{dp}(i)$



Adding $f_i(x)$

3. (7%) Solve the following problem.
   Manaka is a girl who lives in the ocean, and she likes to feed fish. There are a_i fish at position i, for $0 \leq i \leq n$, where 0 represents the westernmost point. Manaka wishes to feed every fish, so she decides to put food at some positions. Placing food at position i would consume Manaka c_i unit of physical power, and the food is enough for any number of fish to eat. Because of ocean currents, the fish could only swim toward the west. Each of the fish will swim to the nearest position that has food. It costs 1 unit of physical power for a fish to swim one unit distance. Since swimming is a strength-consuming work, she wants to minimize the sum of the physical power consumed by her and all of the fish.
   Design an algorithm to solve the problem in $\mathcal{O}(n)$ time.

   Hint: You could use the result from 2 directly, but you need to show that your recursive formula has the same form as equation (1), and all the parameters could be calculated in $\mathcal{O}(1)$ *time.*

Problem (3) (4) is here just to prevent that you think the problems above are quite easy and feel bored.
You don't need to submit these problems. If you do, your solution would be judged, but you won't get any points from these problems.

(3) Quadrangle Inequality Optimization:
Anna and Elsa know that the kids love snowmen, so they decide to build K snowmen. There are n kids who live in position $a_1 < a_2 < \cdots < a_n$, and each of them will go to play with the nearest snowman. Anna and Elsa wonders where to build the K snowmen so that the sum of distances each kids have to walk would be the minimum.

1. Let $\texttt{dp}(k, i)$ represent the minimum total distance to build k snowmen for the first i kids. The recursive formula of this problem has the form

$$\texttt{dp}(k, i) = \min_{0 \le j < i} \texttt{dp}(k-1, j) + w(j, i), \quad 0 \le k \le i \le n$$

where $w(j, i)$ means the minimum distances of the kids from $j+1$ to i if they play with the same snowman. Briefly explain that one of the possible place to build snowman for minimum cost is the position a_t where $t = \lfloor (i+j)/2 \rfloor$ and that $w(j, i) = w(j, i-1) + a_i - a_t$.

2. We could calculate all the $\texttt{dp}$ value in $\mathcal{O}\left(n^2 K\right)$ time easily. Define

$$A(k, i) = \arg\min_{0 \le j < i} \texttt{dp}(k-1, j) + w(j, i)$$

If we know that $A(k, i-1) \le A(k, i) \le A(k+1, i)$, Prove that all the $\texttt{dp}$ values could be calculate in $\mathcal{O}\left(n^2\right)$ time, Assuming that there exists a good dynamic programming order such that Both $A(k, i-1)$ and $A(k+1, i)$ is calculate before $A(k, i)$.

Hint: Calculate

$$\sum_{k=1}^{n} \sum_{i=1}^{n} A(k+1, i) - A(k, i-1) = \sum_{d=i-k=1}^{n} \sum_{i=d+1}^{n} A(i-d+1, i) - A(i-d, i-1)$$

. Expand $\sum_{i=d+1}^{n} A(i-d+1, i) - A(i-d, i-1)$.

*3. Prove that the inequality $A(k, i-1) \le A(k, i) \le A(k+1, i)$ hold for this problem.

Hint: First proof that w satisfied the **quadrangle inequality**

$$w(j, i) + w(j+1, i+1) \le w(j+1, i) + w(j, i+1).$$

Then proof that $\texttt{dp}$ *also satisfied the quadrangle inequality by induction in k*

$$\texttt{dp}(k, i) + \texttt{dp}(k+1, i+1) \le \texttt{dp}(k+1, i) + \texttt{dp}(k, i+1).$$

$A(k, i-1) \le A(k, i)$ could be proved directly using the quadrangle inequality of w. To proof the later, use the one with $\texttt{dp}$.

(4) Divide and Conquer:
After a serious disaster, Kanba and Ringo decide to rebuild a new metro in their home town. The new metro system has n stations, with the route going through station $1, 2, 3, \cdots, n-1, n$ and then back to station 1 (That is, the route is cyclic). Station i is located at (x_i, y_i). Due to some analysis in geology, for each station i, there are m_i posible depth $0 > z_{i,1} > z_{i,2} > \cdots > z_{i,m_i}$ where the station could be located (i.e ., there are total $\sum m_i = m$ candidates). The route would be a straight segment connect consecutive stations, so if they choose to build the station at z_{i,a_i} for each station, the total length would be $d(1,2) + d(2,3) + \cdots + d(n-1,n) + d(n,1)$, where

$$d(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_{i,a_i} - z_{j,a_j})^2}$$

Since the cost is proportional to the total length, they want to choose the best location a_i for each station to minimize the total length.

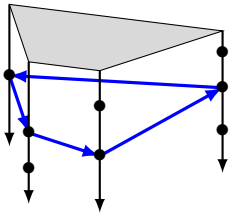


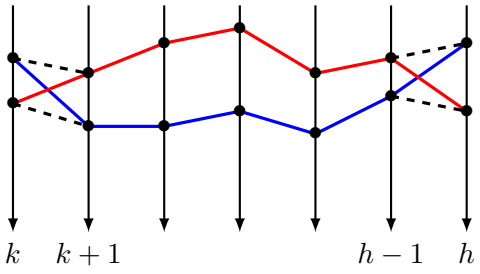
Figure 1: A possible route

We define $p_{i,j}$ to be the point $(x_i, y_i, z_{i,j})$, ans we use a tuple $A = (a_1, a_2, \cdots, a_n)$ to represent a route with points $p_{1,a_1}, p_{2,a_2}, \cdots, p_{n,a_n}$.

1. Let $A = (a_1, a_2, \cdots, a_n)$ and $B = (b_1, b_2, \cdots, b_n)$ be two routes. If there exists $1 \le k < h \le n$ such that $a_k \ge b_k, a_{k+1} < b_{k+1}$ and $a_{h-1} < b_{h-1}, a_h \ge b_h$, (i.e., the two routes cross each other). Briefly explain that if we changes these routes to

$$A' = (a_1, a_2, \cdots, a_k, b_{k+1}, \cdots, b_{h-1}, a_h, \cdots, a_n)$$
$$B' = (b_1, b_2, \cdots, b_k, a_{k+1}, \cdots, a_{h-1}, b_h, \cdots, b_n).$$

Then one of them will be better than before, or the length of both of them will remain the same.



By this lemma, we could safely assume that if A, B are two routes that pass through two different point p, q with minimal length, then these routes will not cross each other.

Hint: Calculate the sum of these two routes and compare to the origin. Use the same idea of the hint from Problem 3.(1) in homework 1.

2. If we know that the best route pass through station 1 at $p_{1,\hat{a}_1}$, give an algorithm to find the minimal route in $\mathcal{O}(m \log m)$ time.

   Of course, this algorithm could also be apply to the case such that the best route pass through a point $p_{k,\hat{a}_k}$ with $k \neq 1$.

   Hint: Let $\texttt{dp}(i,j)$ *be the minimum length start from* $z_{1,\hat{a}_1}$ *and ends at* $z_{i,j}$*. For each station* i*, let* $k = \lfloor m_i/2 \rfloor$*, calculate* $\texttt{dp}(i,k)$ *by going through* $\texttt{dp}(i-1,x)$*. If* $\hat{x}$ *is the one which let* $\texttt{dp}(i,k)$ *be the smallest, Then by 1., when calculating* $\texttt{dp}(i,h)$ *for* $h < k$*, you only need to consider* $\texttt{dp}(i-1,y)$ *for* $y \leq \hat{x}$*.*

*3. By the previous result, if we enumerate through $z_{1,1}, z_{1,2}, \cdots, z_{1,m_1}$, we obtain an $\mathcal{O}\left(m^2 \log_2 m\right)$ solution. But again, by using a divide and conquer method, proof that we could get an $\mathcal{O}\left(m \left(\log_2 m\right)^2\right)$ solution.

   Hint: Choose a station i *and assume the best route would pass through point* $p_{i,k}$ *where* $k = \lfloor m_i/2 \rfloor$*. Calculate the best route* A *in* $\mathcal{O}(m \log_2(m))$ *using the previous problem. Now, since the route wouldn't cross, do a divide and conquer by arranging the points into two groups* P^+, P^-*, such that* P^+ *contains the points that is in* A *or above* A*,* P^- *contains the points that is in* A *or below* A*. solve for the time complexity you'll get* $T = \mathcal{O}(m \log_2(m) \log_2(m_i) + n m_i \log_2(m_i))$*. Take good care for* $n m_i \log_2(m_i)$*. Would choosing* i *with the minimal* m_i *help?*

Problem 4 - Lucky Number (Programming)

Description

(30% + 6% bonus) We all know that digit 7 is lucky and digit 4 is unlucky. For example, you can win a jackpot if you get 777 on a casino slot machine, and elevators will often be missing the 4th floor or any floor whose number contains the digit 4. Even a special term Tetraphobia is created to describe the fear of the digit 4.

In this problem, we say an integer n is a lucky number if:

- The integer n is divisible by 7.
- The decimal representation of n contains at least three digits 7.
- The decimal representation of n contains more digits 7 than digits 4.

For example, 777 and 774746 are lucky numbers; but 7771, 77, and 747474 are not. Can you tell us how many lucky numbers are in range $[l, r]$?

Input Format

The first line contains an integer T indicating the total number of test cases. Each test case contains two integers l, r in one line.

- $1 \leq T \leq 100000$
- $1 \leq l \leq r \leq 10^{18}$

Output Format

For each test case, please output an integer indicating how many lucky numbers are in range $[l, r]$ in one line.

Sample Input

```
3
1 10000
1 1000
1000 10000
```

Sample Output

```
5
1
4
```

Hint

- All lucky numbers in $[1, 10000]$ are 777, 7077, 7707, 7770, 7777.
- There are two bonus tests (3 points each) in this problem. The condition $T \leq 300$ holds for the first 10 tests.