ACM ICPC Kharagpur Regional 2019 Solutions

December 8, 2019

Train or Walk

Problem

There are n cities in a line at $x_1, x_2, \ldots, x_n$. There's a train that goes from city C to D without any intermediate stations, starting at $t = y$ seconds. Currently, $t = 0$. A person wants to go from A to B. He can walk a metre in P seconds, and the train can cover a metre in Q seconds. What is the minimum time to reach B?

Train or Walk

Solution

- If he doesn't board the train and just walks, it takes $P|x_A - x_B|$ seconds.
- If he can reach city C in $\leq Y$ seconds and board the train, then it would take $Y + Q|x_D - x_C| + P|x_B - x_D|$ seconds.
- Return the min over the two possibilities.

Colliding Balls

Problem

There are some balls placed either on positive X-axis (red balls) or on positive Y-axis (blue balls).
At the time $t = 0$, the balls on positive X-axis are thrown towards positive Y-axis and vice versa. The speeds of different balls might be different.
When 2 balls collide, they both disappear. Find the total number of collisions.

Colliding Balls

Solution

- 2 balls indexed i and j respectively can collide only if $\frac{y_j}{u_i} = \frac{x_i}{v_j}$
- This is equivalent to $x_i u_i = y_j v_j$.
- Group all balls according to product of position and speed.
- Let a be the number of red and b be the number of blue balls in some group.
- Observe that the number of collisions from that group is then $\min(a, b)$.

Maximum Diversity

Problem

Given an array A, you are allowed to make k changes to it. Maximize the number of pairs of unequal values.

Maximum Diversity

Solution

- Clearly, it is optimal to choose every replacement as a number not present in the array.
- We have to minimize $\sum f_i^2$, where f_i is the frequency of i.
- Greedy Approach : Always choose the number with maximum frequency and replace it by a number not present.
- Suppose t has the maximum frequency, and it was never changed, whereas another number z was changed.
- If we replace a decrement in frequency of z by a decrement in frequency of t, the objective doesn't increase.

Awkwardness minimization

Problem

Given n boys and m girls, arrange them in a line such that the sum of pairwise distances between each boy and girl is minimized.

Awkwardness minimization

Incorrect Solution 1

Use uniform-ish separation between the boys.

Incorrect Solution 2

Place boys and girls alternatively and place excess boys/girls in the end.

Solution for $n = m$

- It is optimal to place them alternatively. Why?
- Suppose, in an optimal solution, there are $k \geq 2$ boys in the beginning.
- Swap the k^{th} boy with the girl next to him.

Awkwardness minimization

Solution for $n = m$...

- The boy gets distance 1 closer to $n-1$ girls to his right‘, and the girl gets distance 1 further from $n-k$ boys to her right and distance 1 closer to $k-1$ boys to her left.
- The change is $(n-k)-(n-1)-(k-1) = -2(k-1) < 0$.
- The first two students must have different genders.
- Use induction.

Solution for $n > m$

- The first and the last students must both be boys. Why?
- Suppose the first $k \geq 1$ students are girls. Swap the first girl with the first boy.
- Change in objective function is $(m-k)k - k(n-1) = k(m-n-(k-1)) \leq k(m-n) < 0$

Awkwardness minimization

Solution for $n > m$...

- Similarly the last student must also be a boy.
- total cost =
  $m(n + m - 1)$ + cost of middle n + m - 2 students
- Use induction.
- For $n = 7, m = 3$, the solution looks like bbbgbgbgbb

Chef and Diamonds

Problem

There are N chocolates and Q diamonds in a jar. Items are picked one by one at random without replacement. What is the expected number of steps after which all diamonds have been picked?

Chef and Diamonds

Solution

- Lets number the chocolates $1, 2, \ldots N$.
- Let X_i be the event that chocolate i is picked after all the diamonds have been picked.
- Clearly, we are interested in computing $E[N + Q - \sum_{i=1}^{N} X_i]$
- Using symmetry, $E[X_i] = E[X_1]$ for all i.
- $E[X_1] = \frac{1}{Q+1}$, and the answer is $N + Q - \frac{N}{Q+1}$

Analytics Load Jobs

Problem

- There are n events each having two params - the table to write and the size (either 1 or 2 kb).
- Distribute these events into files which would be later written to a database using load jobs.
- A file contains only the events belonging to same table.
- All files in a single load job must write to the same table.
- Each load jobs can write at most K files.
- You can make at max J load jobs.
- The task is to distribute events in the files as evenly as possible, i.e. the file sizes shouldn't be too large. Formally, minimize the maximum size of any file.

Analytics Load Jobs

Solution

- Whenever you have to minimize the maximum value or reverse of it (i.e. maximize the minimum value), think binary search.
- The idea here is similar. Binary search over the answer.
- In binary search predicate, you would answer whether it is possible to perform load jobs with the maximum file size being $\leq mid$?
  - If yes, then try with decreasing file size, i.e. mid
  - Otherwise, try with increasing mid.

Analytics Load Jobs

Solution

Checking the predicate

- It can be done greedily. Collect all the events that must be written to same table. We know that maximum file size is *mid*. How compactly can we pack these events into the least number of files?
- *Greedy strategy*: Fit as many 2's first as possible. If there is still space left, fill it with ones. The idea is that 1's are easier to fit than 2's.
- Thus, if the total number of files needed are $> J * K$, then it's not possible with current *mid*, try higher mid.
- Otherwise, try with lower *mid*?

Special Graph Construction

Problem

A special graph is one with the following properties:

- It has ≤ 1000 vertices.
- It is a simple connected 3-regular graph.
- It has exactly k bridge edges.

Define $f(G)$ to be the minimum number of edges to be removed from the G to make it bipartite. Find a special graph with minimum possible f value.

Special Graph Construction

$f(G) \geq 2$ for all $k \geq 1$

- Clearly, n must be even as $3n = 2m$. Consider a bridge edge e. On removing e, we must be left with two components.
- In each component all but one vertices have degree 3 and one vertex has degree 2.
- Consider one such component. We claim that it can't be bipartite.
- Assume it is bipartite with a nodes on the left all of degree 3, and b nodes on the right with $b-1$ nodes of degree 3 and one vertex of degree 2.
- $3a = 3(b-1) + 2$, a contradiction.
- $f(G) \geq 2$ as we have to remove atleast one edge from each component.

Special Graph Construction

Solution...

- If $k = 0$, $K_{3,3}$ is a valid solution. (Also used in sample IO).
- If $k \geq 1$, graphs with $f(G) = 2$ always exist. As proved already, this is the best possible.

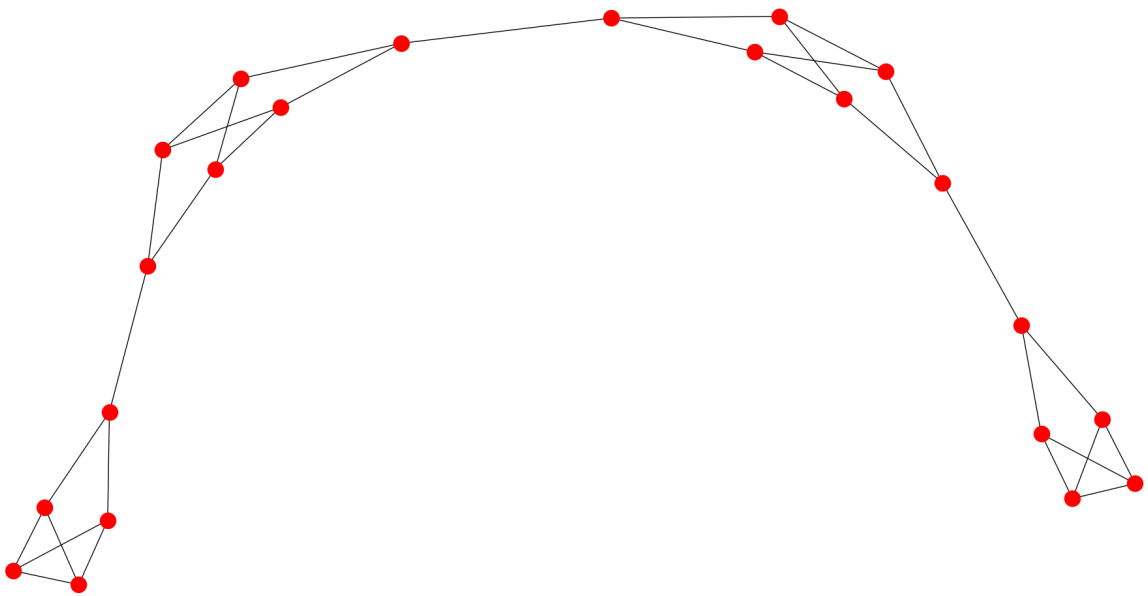

Figure: Example for k = 3

Special Graph Construction

Solution...

- The above example can easily be generalized to any $k \geq 1$
- It uses about $6k$ vertices. Since $k \leq 10^4$, this works.
- It has $f(G) = 2$ which is the minimum possible.

Recover Array

Problem

Ask $\leq 9 \times 10^4$ subarray sum queries to recover a binary array of size 10^5.

Recover Array

Solution

- Let $n = 10^5$. Divide the array into $\frac{n}{3}$ chunks of size 3 each.
- Can we recover a chunk of size 3 in < 3 queries on average?
  - Find the sum of the chunk.
  - If the sum is 0 or 3, we know that it is all zeroes or all ones respectively.
  - Say the sum is 1, then we need to find the only 1. If the sum is 2 we can similarly search for the only 0.
  - Lets do a randomized search for it.
  - Query a random index, If it has 1, we are done. Else, query another index. If it has 1, we are done, else the only unasked index must have 1.

Recover Array

Analysis

- If sum is 0 or 3 we use only 1 query, which is good enough.
- If we succeed in the first search, we use 2 queries. This happens with probability $\frac{1}{3}$.
- Otherwise, we use 3 queries, with probability $\frac{2}{3}$
- Expected number of queries per chunk is therefore $\frac{1}{3} \times 2 + \frac{2}{3} \times 3 = \frac{8}{3}$.
- Overall expected number of queries is therefore $\frac{8}{3} \times \frac{n}{3} = \frac{8n}{9}$

Analysis...

- We are allowed to use $\frac{9n}{10}$ queries.
- Let x_i be a random variable denoting the number of queries saved (as opposed to using 3 queries) in the i^{th} chunk.
- x_i are all independent. $x_i = 0$ with probability $\frac{2}{3}$, and $x_i = 1$ with probability $\frac{1}{3}$.
- Let $X = \sum x_i$. Clearly, $\mu = E(X) = \frac{n}{9}$. Using chernoff bounds,
- The failure probability

$$= Pr[X < \tfrac{n}{10}] = \Pr\left[X < \left(1 - \frac{1}{10}\right)\frac{n}{9}\right] \leq e^{-\frac{n}{1800}} < 10^{-20}$$

Colorful Balloons

Problem

The cost of an array is defined as $\sum f_r^k$, where f_r is the frequency of r in the array. Given an array, find the sum of costs of all its subarrays.

Colorful Balloons

Solution

- For each r, we find the contribution of r.
- Let r be present at positions $p_1, p_2, \cdots p_c$. Let $p_0 = 0, p_{c+1} = n + 1$
- The number of subarrays in which r is present exactly m times is $y_m = \sum_{i=1}^{c-m+1} (p_i - p_{i-1})(p_{i+m} - p_{i+m-1})$
- Let $P = \sum (p_i - p_{i-1}) x^{-i}$, $Q = \sum (p_j - p_{j-1}) x^j$.
- Clearly, $y_m =$ coefficient of x^m in PQ.
- The contribution of r is then $\sum_m y_m m^k$

Analysis

- For each r, the polynomials P and Q have size c_r each where c_r is the frequency of r in the array.
- Using FFT to multiply P and Q takes $O(c_r \log c_r)$ time.
- The sum of coefficients of PQ is n^2, which is $\leq 4 \times 10^{10}$.
- Since the coefficients don't get too large, we can use FFT with doubles.
- Calculating $\sum_m y_m m^k$ takes time $O(c_r \log k)$
- Overall complexity is $\sum O(c_r \log(c_r) + c_r \log k) = O(n \log(nk))$

Minimum Variance

Problem

Given n sets $S_1, S_2, \ldots, S_n$, choose $x_i \in S_i$ for all $1 \leq i \leq n$, such that the variance of $\{x_1, x_2, \ldots, x_n\}$ is minimized.

Minimum Variance

Solution

- It is well known (and easy to prove) that the value of $\sum_{i=1}^{n}(x_i - t)^2$ is minimized at $t = \mu$
- We have to find

$$\min_x \sum_{i=1}^{n}(x_i - \mu)^2 = \min_x \left(\min_t \sum_{i=1}^{n}(x_i - t)^2 \right)$$

- Swapping the order of min, we have

$$\min_t \left(\min_x \sum_{i=1}^{n}(x_i - t)^2 \right)$$

Minimum Variance

Solution...

- Clearly, for a fixed t, for each i, we must choose $x_i \in S_i$ closest to t.
- In a given sorted set, the choice for the closest element to t changes only at mid of any two consecutive numbers in the set.
- So, when we iterate t from $-\infty$ to ∞, the optimal vector x changes only $O(\sum |S_i|)$ times.
- Find the value for each possible optimal vector.
- We can maintain $\sum x_i$, and $\sum x_i^2$ in $O(1)$ per change, and compare $n \sum x_i^2 - (\sum x_i)^2$ after each change.

Minimum Variance

Analysis

- We have $O(\sum |S_i|)$ changes, and each change is processed in $O(1)$, so the overall complexity is $O(\sum |S_i|)$

Direct Segments

Problem

Given n segments. For each segment assign it a direction along the segment, such that no two segments ever intersect if they start moving in the assigned direction with equal speeds.

Direct Segments

Solution

- Let L_i be the infinite line defined by extending the i^{th} segment.
- First choose a default direction for each segment.
- For a segment i, let x_i to be 1 if it chooses the default direction and 0 if it chooses the opposite.
- Consider two segments i and j. Given their directions, do they intersect?
- The time range of intersection (maybe empty) can be easily found by finding intersection of time ranges when the two ends of segment i are on opposite sides of L_j and vice versa

Direct Segments

Solution...

- If using these directions, the two segments intersect, add a condition that atleast one should have a different direction than the one considered.
- Solve this instance of 2SAT.

Analysis

- For every pair and the pairwise directions, intersection can be found in $O(1)$. Overall, this takes time $O(n^2)$
- We have n variables and $O(n^2)$ clauses in the 2SAT instance formed. This also takes $O(n^2)$ to solve.
- Overall complexity : $O(n^2)$

Substring Matching

Problem

Find the longest substring of a string T that can be formed by concatenating a subset of strings $S = \{S_1, S_2, \ldots S_n\}$ in a non decreasing order of length.

Substring Matching

Solution

- Let f(i, l) be the right end of the longest string staring at position i in T that can be made by concatenating some strings with length $\geq l$ in non decreasing order of length.
- The answer would be $\max f(i, 1) - i + 1$
- Clearly, $f(i, l) = \max_{0 \leq r \leq R_i} f(i + rl, l + 1)$, where R_i is the maximum number of strings of length l that can be put next to index i matching the corresponding length l substrings in T.
- There are $O(\sqrt{N})$ different lengths possible.

Substring Matching

Solution...

- But this would still take $O(N^2)$ time, if we naively take the maximum over all possible $r \leq R$, for example when S consists only of a, and $T = aaaaa\ldots$.
- Let's divide the positions into groups according to $i \bmod l$. This makes sense as the recurrence only depends on states in the same group.
- Let H_l be the multiset of hashes of strings in S of length l.
- If we had to find R_i naively, we would keep searching for hashes of length l substrings $(i \ldots i+l-1,\ i+l,\ i+2l-1,\ \ldots)$ in H_l and removing them from H_l until we can do so.
- This process can be sped up using 2 pointers approach.

Substring Matching

Solution...

- Clearly, $R_{i+l} \geq R_i$
- To find R_{i+l}, add back the hash of string $i \ldots i+l-1$ to H_l and increase the right end until possible.
- Also maintain a multiset of dp values of the current window, so you use the recurrence.

Analysis

- For a length l, each group takes $O(\frac{N}{l}) \log(\frac{N}{l})$ time as the right index of the window can only move $O(\frac{N}{l})$ times.
- There are l groups, and hence each distinct length takes time $O(N \log N)$
- Overall complexity is $O(N\sqrt{N} \log N)$ with a small constant factor.