第 6 回 ドワンゴからの挑戦状 予選 解説
(Editorial for Dwango Programming Contest VI Preliminary Round)

ドワンゴ競技プログラミング同好会

https://atcoder.jp/contests/dwacon6th-prelims/

2020/1/11 (土)

For International Readers: English editorial starts on page 6.

A: Falling Asleep

N 個の曲のうち、$s_i = X$ なる i を探し、$\sum_{j=i+1}^{N} t_j$ を求めればよいです。これは文字列比較にかかる時間を無視すると、$O(N)$ で実行可能です。

C++ での実装例: https://atcoder.jp/contests/dwacon6th-prelims/submissions/9422972

B: Fusing Slimes

$(N-1)!$ を最後にかけることにすると、求めるものはそれぞれのスライムが移動する距離の総和の期待値です。それぞれのスライムが移動する距離を求める代わりに、$1 \leq i < N$ を満たす整数 i について、x_i, x_{i+1} 間を各スライムが移動する確率を考えます。

以降、左から j 番目のスライムのことを単にスライム j、スライム j が x_i, x_{i+1} 間を移動する確率を $p_{j,i}$ と書くことにします。それぞれのスライムが移動する距離の総和の期待値は $\sum_{i=1}^{N-1}(x_{i+1}-x_i)\sum_{j=1}^{i} p_{j,i}$ と表せます。

$p_{j,i}$ の具体的な値は以下のように求められます。

- スライム i は必ず x_i, x_{i+1} 間を移動するので、$p_{i,i} = 1$ です。
- スライム $i-1$ が x_i, x_{i+1} 間を移動するためには、先にスライム i が選ばれる必要があります。よって、$p_{i-1,i} = 1/2$ です。
- スライム $i-2$ が x_i, x_{i+1} 間を移動するためには、先にスライム $i-1, i$ が選ばれる必要があります。よって、$p_{i-2,i} = 1/3$ です。
- より一般のスライム $i-j$ について、$p_{i-j,i} = 1/(j+1)$ が成立します

よって、$1 \leq i < N$ を満たす整数 i について $(x_{i+1}-x_i)\sum_{j=1}^{i}\frac{1}{j}$ を計算すればよく、これは適切に前計算をすれば $O(N)$ で求められます。

C: Cookie Distribution

うれしさの期待値に $\binom{N}{a_1} \times \binom{N}{a_2} \times \ldots \times \binom{N}{a_K}$ をかけた値、というのはありうるクッキーの配り方の全てについてうれしさを計算し、その総和を取った値と等しいです。以降はこれを求めることを考えます。

全ての配り方についてうれしさを計算することは、各子供が受け取ったクッキーの数を管理する必要があり困難です。ここで、うれしさがそれぞれの子供が持っているクッキーから 1 枚を選ぶ方法の個数と言い換えられることが重要です (子供がクッキーを受け取らなかった場合のうれしさは 0 のため、考慮する必要はありません)。

それぞれの子供について選ばれたクッキーが何日目に受け取ったクッキーなのかを固定したときに、ありうるクッキーの配り方が何通り存在するかを考えてみることにします。これは、それぞれの日について子供たちに選ばれなかったクッキーを残りの子供に配る方法の総乗であるので、i 日目に配られたクッキーが子供達に選ばれた個数を x_i として、$\prod_{i=1}^{K} \binom{N-x_i}{a_i-x_i}$ 通りなことがわかります。さらに、$x_1, x_2, \ldots, x_K$ を固定したとき、このようなクッキーの選ばれ方は $\frac{N!}{\prod_{i=1}^{K} x_i!}$ 通りあります。よって、ありうる全ての $x_1, x_2, \ldots, x_K$ について、$N! \prod_{i=1}^{K} \frac{\binom{N-x_i}{a_i-x_i}}{x_i!}$ を計算し、その総和を取った値が答えです。

各 i について、x_i の値だけが係数に関与することから $dp(n, m)$ を $\sum_{i=1}^{n} x_i = m$ なる $x_1, x_2, \ldots, x_n$ について、$N! \prod_{i=1}^{n} \frac{\binom{N-x_i}{a_i-x_i}}{x_i!}$ を計算した値の総和、とした DP でこの問題を解くことができます。状態数が $O(KN)$ で遷移が $O(N)$ なので、全体として $O(KN^2)$ の計算量で動作し十分高速です。

D: Arrangement

この問題は、各頂点の出次数が 1 であるようなグラフ G が与えられるので、G の補グラフにおける辞書順最小のハミルトンパスを求める問題です。一般のグラフにおいてはハミルトンパスを求めるのは困難ですが、今回与えられるグラフについては以下の補題が成り立つことが重要です。

> $V \subset \{1, 2, \ldots, N\}$ なる V による誘導部分グラフ G' について、以下の条件が G' の補グラフに s を始点とするようなハミルトンパスが存在するための必要十分条件である。
>
> - $|V| \neq 3$ のとき、s 以外の頂点のうち、入次数が $|V| - 1$ であるようなものが存在しない
> - $|V| = 3$ とき、上記の条件に加えて s を取り除いたとき、s 以外の頂点 u, v について $u \to v, v \to u,$ の少なくともいずれかが存在しない

必要性は明らかです。十分性を数学的帰納法を用いて示します。$|V| = 1, 2$ のときは明らかです。$|V| = 3$ のとき、ありうるグラフは 27 通りありますが、ハミルトンパスが存在するものは上記の 2 つの条件を満たすものに限られます。

$|V| > 3$ のときを考えます。$V' = V \setminus \{s\}$ による誘導部分グラフを G'' とします。まず、G' において、s 以外の頂点の入次数が全て $|V| - 2$ 未満の場合を考えます。このとき、G'' の頂点はいずれも入次数が $|V'| - 2$ 以下となります。$|V| = 4$ のとき、 G'' において、ハミルトンパスの始点としてありうる頂点が少なくとも 2 つ以上存在します。また、$|V| > 4$ においては、任意の頂点が G'' のハミルトンパスの始点になりえます。s から移動不可能な頂点は 1 つに限られるため、s を始点とするハミルトンパスが必ず存在します。

次に、G' において、s 以外の頂点であって、入次数が $|V| - 2$ であるような頂点 v が存在する場合を考えます。$s \to v$ なる辺が存在しないなら、v を始点とするハミルトンパスが G'' に存在します。よって、$s \to v \to \ldots$ というハミルトンパスが存在します。一方、$s \to v$ なる辺が存在するなら、v 以外の頂点であって $t \to v$ なる辺が存在しない頂点 t を始点とするハミルトンパスが G'' に存在します。よって、$s \to t \to v \to \ldots$ というハミルトンパスが存在します。以上より、いずれの場合も s を始点とするハミルトンパスが存在することが示されました。

以上の補題を利用して、「ハミルトンパスが存在する」という条件を保ったまま訪問可能な頂点のうち番号最小のものを選んで移動し続けることで、辞書順最小の並びを求めることが可能です。適切に実装すると $O(N \log N)$ で実行可能で、十分高速です。

E: Span Covering

1 個の区間を置くごとに、ある 2 つの区間について共通部分があるなら和集合を取って 1 つの区間にする、という操作を可能な限り繰り返すことにします。このように操作を行った結果、最終的に共通部分を持たないような k 個の半開区間 $[l_1, r_1), [l_2, r_2), \ldots, [l_k, r_k)$ がある状態になります。

被覆されていない点がないような点がないとき、$\sum_{i=1}^{k} r_i - l_i = X$ が成立しています。被覆されていない点がないような置き方を数える代わりに、和集合を取った結果として $L_i = r_i - l_i$ として $\sum_{i=1}^{k} L_i = X$ を満たす k 個の半開区間になるような区間の置き方の個数を数えることにします。

L は $L_1 \geq L_2 \geq \ldots \geq L_N$ としても一般性を失いません。以降、L は降順であることを仮定します。すると、ニワンゴ君が長さ l の区間を置いたとき以下のいずれかが生じることがわかります。

- 長さ l の区間が追加される
- ある 1 つの区間と併合される
- ある 2 つの区間と併合される

上記のいずれかしか生じない、特に 3 つ以上の区間と併合され 1 つの区間になる場合が存在しないことは、L が降順に並んでいることから容易に示せます。区間が実際に置かれている位置はあまり重要ではないため、区間の個数 k とその長さ $L_1, L_2, \ldots, L_k$ だけを管理し、上記のそれぞれの場合についてそれぞれ考えることにします。

長さ l の区間が追加される場合

L_{k+1} として、l を単に追加すればよいです。そのような置き方は 1 通りです。

ある 1 つの区間と併合される場合

k 個の区間から 1 つ選び (選ばれた区間を i とします) 併合されたとき、併合後の区間の長さが増加しないような置き方は $L_i - (l-1)$ 通りです。また、併合後の区間の長さが $L_i + x (1 \leq x < l)$ となるような置き方は左右のどちら側を伸ばすかで 2 通りあります。

ある 2 つの区間と併合される場合

k 個の区間から 2 つ選び (併合前に左側にある区間を i、右側にある区間を j とします) 併合されたとき、併合後の区間の長さが $L_i + L_j + x (0 \leq x < l-1)$ となるような置き方は $(l-1-x)$ 通りあります。

それぞれの場合について、何が起こるかはわかりました。特に、ある 1 つの区間と併合される場合以外は区間の長さ L_i が影響していないことがわかりました。さらに 区間 $1, 2, \ldots, k$ について総和を取ると、$-k(l-1) + \sum_{i=1}^{k} L_i$ 通りだけ置き方が存在し、区間の長さの総和だけがわかればよいことがいえます。以上のことから、$dp(i, n, x) = i$ 個の区間を置いたとき、n 個の区間があり、その長さの総和が x となるような区間の置き方の総数、という DP でこの問題を解くことができます。愚直に実装すると状態数が $O(N^2X)$ 遷移が $O(L_i)$ となって計算量は $O(N^2X^2)$ となります。

ここで、i 番目の区間を置くとき、$n \leq X/L_i$ が成立することを利用すると、着目すべき状態が $O(X^2/L_i)$ 個となって、計算量は $O(NX^2)$ となります。これは十分高速に動作します。余談ですが、遷移が $O(L_i)$ となるのは区間が併合され、区間の長さの総和が増加する場合に限られます。これを累積和を用いてまとめることで $O(N^2X)$ で解くこともできます。

Editorial for Dwango Programming Contest VI Preliminary Round

Dwango Competitive Programming Club

https://atcoder.jp/contests/dwacon6th-prelims/

Saturday, January 11, 2020

A: Falling Asleep

We should find i such that $s_i = X$ and find $\sum_{j=i+1}^{N} t_j$, which takes $O(N)$ time ignoring the time for string comparison.

Sample C++ implementation: https://atcoder.jp/contests/dwacon6th-prelims/submissions/9422972

B: Fusing Slimes

Let us find the expected value itself and multiply it by $(N-1)!$ later. Instead of finding the distance each slime travels, for each i such that $1 \leq i < N$, let us find the probability that each slime covers the section from x_i to x_{i+1}. (When two slimes get fused, assume that the slime from the left disappears and the right slime remains.)

Below, we call the j-th slime from the left just Slime j, and let $p_{j,i}$ be the probability that Slime j covers the section from x_i to x_{i+1}. We can represent the total distance covered by the slimes as $\sum_{i=1}^{N-1}(x_{i+1}-x_i)\sum_{j=1}^{i} p_{j,i}$.

The values $p_{j,i}$ can be found as follows:

- Slime i always covers the section from x_i to x_{i+1}, so $p_{i,i} = 1$.
- In order for Slime $i-1$ to cover the section from x_i to x_{i+1}, Slime i needs to be chosen by Niwango earlier than Slime $i-1$, so $p_{i,i} = 1/2$.
- In order for Slime $i-2$ to cover the section from x_i to x_{i+1}, both Slime $i-1$ and i need to be chosen earlier than Slime $i-2$, so $p_{i,i} = 1/3$.
- Generally, for Slime $i-j$, $p_{i-j,i} = 1/(j+1)$ holds.

Thus, we can found the answer by computing $(x_{i+1}-x_i)\sum_{j=1}^{i}\frac{1}{j}$ for each i such that $1 \leq i < N$, which can be done in $O(N)$ time with the help of pre-computation.

C: Cookie Distribution

The expected happiness multiplied by $\binom{N}{a_1} \times \binom{N}{a_2} \times \ldots \times \binom{N}{a_K}$ is equal to the sum of the happiness over all possible ways to distribute cookies. Let us find this value.

It is hard to directly compute the happiness, since it requires us to maintain the number of cookies received by each child. Thus, it is critical to notice that the happiness can be rephrased as the number of ways to choose, for each child, one cookie from the cookies received by that child. (If some child receives no cookie, the happiness is 0, so we can ignore such a case.)

Let us consider the number of possible ways to distribute cookies when we fix, for each child, the day when that child receives the "chosen" cookie. This is the product of the numbers of ways over K days to distribute "unchosen" cookies to the other children, so there are $\prod_{i=1}^{K} \binom{N-x_i}{a_i-x_i}$ such ways, where x_i is the number of chosen cookies on Day i. Additionally, when $x_1, x_2, \ldots, x_K$ are fixed, there are $\frac{N!}{\prod_{i=1}^{K} x_i!}$ ways to decide the day for each child to receive the chosen cookie. Therefore, the answer is the sum of $N! \prod_{i=1}^{K} \frac{\binom{N-x_i}{a_i-x_i}}{x_i!}$ over all possible sequences $x_1, x_2, \ldots, x_K$.

For each i, only x_i contributes to the coefficient, so this can be solved with DP by letting $dp(n, m)$ be the sum of $N! \prod_{i=1}^{n} \frac{\binom{N-x_i}{a_i-x_i}}{x_i!}$ over all sequences $x_1, x_2, \ldots, x_n$ such that $\sum_{i=1}^{n} x_i = m$. We have $O(KN)$ states and $O(N)$ transitions for each of them, so the complexity is $O(KN^2)$, which works fast enough.

D: Arrangement

The problem is as follows: given a graph G where the out-degree of each vertex is 1, find the lexicographically smallest Hamiltonian path of the complement graph of G. For a general graph, it is hard to find its Hamiltonian path, but for a graph given this time, we have the following important lemma:

For the induced subgraph G' by V such that $V \subset \{1, 2, \ldots, N\}$, there exists a Hamiltonian path in the complement graph of G' starting at s if and only if the following condition on G' is met:

- When $|V| \neq 3$: except for s, G' has no vertex whose in-degree is $|V| - 1$.
- When $|V| = 3$: the above condition must hold. Additionally, for the two vertices u and v other than s, at least one of the following edges is absent: $u \to v$ and $v \to u$.

The necessity of this condition is obvious. Let us show the sufficiency by induction. For $|V| = 1, 2$, it is obvious. For $|V| = 3$, there are 27 possible graphs. Among them, only the ones satisfying the above condition results in its complement graph having a Hamiltonian path.

We will consider the case $|V| > 3$. Let G'' be the induced graph by $V' = V \setminus \{s\}$. Let us first consider when, in G', the out-degree of every vertex except s is less than $|V| - 2$. In this situation, the in-degrees of all the vertices in G'' are at most $|V'| - 2$. When $|V| = 4$, G'' has at least two vertices that can be the starting point of a Hamiltonian path. When $|V| > 4$, all the vertices in G'' can be the starting point of its Hamiltonian path. Since there is only one vertex that cannot be directly reached from s, there always exists a Hamiltonian path starting at s.

Now, let us first consider when G' has a vertex v with the out-degree of $|V| - 2$. If the edge $s \to v$ is absent, the complement graph of G'' has a Hamiltonian path starting at v, so the complement graph of G' has a Hamiltonian path of the form $s \to v \to \ldots$. If the edge $s \to v$ is present, the complement graph of G'' has a Hamiltonian path starting at a vertex t ($\neq v$) such that the edge $t \to v$ is absent, so the complement graph of G' has a Hamiltonian path of the form $s \to t \to v \to \ldots$. Therefore, there always exists a Hamiltonian path starting at s.

From this lemma, we can find the lexicographically smallest Hamiltonian path by repeatedly moving to the vertex with the smallest index that can be visited while keeping the condition for a Hamiltonian path to exist. When properly implemented, this can be done in $O(N \log N)$ time, which is fast enough.

E: Span Covering

Each time we put a segment, let us merge pairs of segments with non-empty intersections. After this merge, let us say we have k non-overlapping half-open intervals $[l_1, r_1), [l_2, r_2), \ldots, [l_k, r_k)$.

If no point is left uncovered, $\sum_{i=1}^{k} r_i - l_i = X$. Thus, let us count the ways to put segments that, after the merge, result in k half-open intervals such that $\sum_{i=1}^{k} L_i = X$, where $L_i = r_i - l_i$.

Without loss of generality, we assume $L_1 \geq L_2 \geq \ldots \geq L_N$. When we put a segment of length l, one of the following happens:

- A new interval of length l is added.
- The segment gets merged with one existing interval.
- The segment gets merged with two existing intervals.

It can be easily shown that nothing else happens (especially, the segment does not get merged with three or more existing intervals), from the fact that L is non-increasing. The positions of the intervals do not matter much, so let us maintain only the number of intervals k and their lengths $L_1, L_2, \ldots, L_k$, and analyze each of the cases above:

The case "a new interval of length l is added":

There is one way to do this: just add l as L_{k+1}.

The case "the segment gets merged with one existing interval":

Let us say the segment is merged with Interval i. There are $L_i - (l-1)$ ways to put the segment that does not increase the length of the new interval after the merge. Additionally, for each x $(1 \leq x < l)$, there are two ways to put the segment so that the new interval after the merge has the length $L_i + x$ (the interval can be extended to either left or right).

The case "the segment gets merged with two existing intervals":

Let us say the segment is merged with Interval i (left) and Interval j (right). For each x $(0 \leq x < l-1)$, there are $(l-1-x)$ ways to put the interval so that the new interval after the merge has the length $L_i + L_j + x$.

We can see from above that the length of the segment L_i does not matter except in the case where it gets merged with one existing interval. Additionally, if we sum up those number of ways over the segments $1, 2, \ldots, k$, there are $-k(l-1) + \sum_{i=1}^{k} L_i$ ways in total. That is, we only need to maintain the sum of lengths of the intervals. Therefore, the problem can be solved with DP by letting $dp(i, n, x) =$ (the number of ways to put i segments that are merged into n intervals with the total length of x). When naively implemented, we have $O(N^2X)$ states and $O(L_i)$ transitions for each of them, resulting in complexity of $O(N^2X^2)$.

Here, let us use the fact that we have $n \leq X/L_i$ when placing the i-th segment. Now we have $O(X^2/L_i)$ states, which make the complexity $O(NX^2)$. This works fast enough. We can also solve the problem in $O(N^2X)$ time, by using prefix sums to put together the $O(L_i)$ transitions, which only appears when intervals get merged and their total length increases as a result.