「みんなのプロコン」解説

writer : yutaka1999

2017 年 3 月 4 日

A : Yahoo

与えられる文字列の長さは 5 なので、この文字列を並び替える方法は高々 $5! = 120$ 通りである。なのでこれをすべて試しても解くことはできるが、今回は純粋に与えられる文字列をアルファベット順に sort したものと、yahoo を sort した文字列が一致するかどうかを調べるだけで解くことができる。

B : オークション

各日にちに購入する商品の番号を　$i_1, i_2, ..., i_K$ とする。このとき、k 日目の商品の値段は $A_{i_k} + k - 1$ であるので、全体での合計は $\sum A_{i_k} + \frac{K*(K-1)}{2}$ となる。よって、各商品を買う順番は関係なく、A_1 から A_N までで金額が低い方から K コを買うのが最適だと分かる。これは、sort することで、$O(N \log N)$ で解くことができるのでよい。

C : 検索

検索ワード T が満たすべき条件は次のようにまとめられる。

- サイト v を検索結果に出したいならば、S_v が T を含む。
- サイト v を検索結果に出したくないならば、S_v が T を含まない。

ここで、検索結果に出したいサイトを一つ考え、それを r とする。このとき、先の条件は次のように言い換えられる。

- S_r が T を含む。
- サイト v を検索結果に出したいならば、S_v と S_r の共通接頭辞が T を含む。
- サイト v を検索結果に出したくないならば、S_v と S_r の共通接頭辞が T を含む。

よって、r 以外の各サイト v について、S_v と S_r の共通接頭辞の長さを L_v とすると、条件はさらに以下のように言い換えられる。

- S_r が T を含む。
- サイト v を検索結果に出したいならば、$|L_v| \geq |T|$
- サイト v を検索結果に出したくないならば、$|L_v| < |T|$

よって、各 L_v をあらかじめ求めておくことで、検索結果に出したいサイトでの L の値の最小値 m と検索結果に出したくないサイトでの L の値の最大値 M を求めることができ、T の条件は

- S_r が T を含む。
- $M < |T| \leq m$

となる。よって、このような条件を満たす T が存在するかどうかは $M < m$ かどうかで判定でき、このとき、S_r の長さ $M+1$ の接頭辞が求める T となる。

さて、計算量を見積もると、L の値が各文字列に対して求められれば良いことになるが、サイト v に対して、$L_v \leq |S_v|$ より、何文字まで一致するか愚直に試しても、合計で高々 $\sum |S_v|$ 回しか計算しなくてよい。よって、$\sum |S_v| \leq 10^5$ より、十分高速に解くことができる。

D：工場

まず、いくつかの注文がたまっている状態で、経営質問 D に答えることだけを考える。i 日目にたまっている注文の A の個数の和を A_i と置くことにすると、以下のような 貪欲法で解くことができる。

- $i = 1 \sim D$ と順番に動かし、残り個数 Z を持つ。最初は $Z = 0$ である。
- i 日目の生産：$Z \mapsto Z + K$ とする。
- i 日目の注文：$Z \mapsto \max(0, Z - A_i)$ とする。

つまり、各日 i に対して、$Z \mapsto \max(0, Z + K - A_i)$ を行い、D 日目まで繰り返せば、$KD - Z$ が求める答えとなる。簡単のため、$B_i = K - A_i$ と置くことで、$Z \mapsto \max(0, Z + B_i)$ という操作を行うとしてよい。

さて、次はクエリがあるときを考える。注文追加のクエリは B_i の変化、経営質問のクエリはある D 日目まで先の関数に通したとき、出てきた Z の値を求めるクエリとなるから、これらのクエリは segment tree を用いることで高速に処理することができる。具体的には以下のようにする。

- segment tree の各ノード $[L, R]$ に、Z が入ってきたらその区間を出た後は $\max(Y_{L,R}, Z + X_{L,R})$ となるというような値 X, Y を対応させる。

このように $X_{L,R}$ を定義できることは次のように示せる。$[L, \frac{L+R}{2}]$ に対応する X, Y を a, c 、$[\frac{L+R}{2}, R]$ に対応する X, Y を b, d とする。このとき、L 日目で Z ならば、$\frac{L+R}{2}$ 日目では $\max(Z + a, c)$、R 日目では $\max(\max(Z + a, c) + b, d) = \max(Z + a + b, \max(b + c, d))$ となり、$[L, R]$ に対応する X, Y を $a + b, \max(b + c, d)$ とすることができる。長さ 1 の区間では注文の処理より明らかに X, Y が対応するから、以上のようにして X, Y を長さが短いものから順に対応づけることができる。

さて、この segment tree を使って実際にクエリを処理することを考える。$M = \max D$ とする。このとき、注文クエリに対しては、変化する B_i に対して、その日を含む区間すべてを長さの短いものから順に更新すればよいので、$O(\log M)$ で解くことができる。さらに、経営クエリも同様に普段の segment tree のように解くことができる。よって、毎クエリあたり $O(\log M)$ で解くことができるが、$M = 10^9$ と大きいため、このままでは MLE してしまう。しかし、前計算をして、D として取りうる値だけに座標圧縮しておくことで、メモリも $O(Q)$ に収めることができ、時間計算量も毎クエリあたり $O(\log Q)$ で解くことができる。よって、これで十分高速に解くことができる。

E：遊園地

まず、アトラクション i の後にアトラクション j を訪れることができるならば i から j に辺を張る、というようにして作ったグラフを求めれば、問題が解けることを示す。

まず、この有向グラフを強連結成分分解（SCC）する。このとき、高橋君の乗ることができるアトラクションの集合は、SCC した後ではパスとなる。よって、このグラフを求めることができれば、SCC した後のパスが含む頂点数の最大を求めればよいことになる。SCC をすると、強連結成分は DAG をなすので、これは DP をすることで $O(N)$ で求められる。

よって、このグラフを効率よく作ることを考える。題意より、アトラクション i を訪れた後にアトラクション j に訪れることができる条件は、$L_j \leq R_i - |i-j|$ となる。よって、この条件は以下のように言い換えられる。

- $i > j$ のとき、$L_j - j \leq R_i - i$
- $i < j$ のとき、$L_j + j \leq R_i + i$

対称性があるので、$A_i = L_i - i, B_i = R_i - i$ として、$j < i$ かつ $A_j \leq B_i$ ならば辺を張る、という操作ができればよいことになる。つまり、以下のように辺を張ればよいことになる。

- $(A_i, 0), (B_i, 1)$ を sort し、小さい順に走査する。
- $(A_i, 0)$ を対象とするとき、i を追加する。
- $(B_i, 1)$ を対象とするとき、$[0, i)$ の間の追加された頂点全てに辺を張る。

愚直にやると明らかに辺の数が大きすぎるが、segment tree を用いることで、効率的に解くことができる。つまり、

- v に頂点を追加する。
- 区間 $[L, R]$ に追加された頂点すべてに辺を張る。

という操作ができればよい。これは、segment tree の各ノードに対して、グラフの頂点としての情報を持たせればよい。頂点を追加するときには、その点を含むノードすべての頂点を作り替え、前の頂点と v に辺を張ればよく、区間の頂点に辺を張るときには、普段の segment tree と同様にすればよい。

これでは、頂点追加により、毎回 $O(\log N)$ 頂点追加されるので、全体として、$O(N \log N)$ 頂点できることになる。よって、直接的ではないが、求めたいグラフで辺がある頂点の間には、有向辺のパスが存在するような $O(N \log N)$ 頂点 $O(N \log N)$ 辺グラフを構成することができる。よって、先と同様にすることで、$O(N \log N)$ で問題を解くことができる。