AtCoder 練習帳 6

2022-07-30 (Sat)

しばらく ARC は出るとしても Unrated で出ようと決めていたので、それで。とか言いつつ、ひょっとしたら 3 問解けちゃったりするかも？とか期待していたりしたけど、全然でした。0 完。Unrated でよかったよ。

ARC145A - AB Palindrome

解説読んだらそのまんまなんだけど、コンテスト中にちゃんと辿り着けなかったことへの反省も込めて、自分でも解き方を整理しておく。

文字列が B から始まる 3 文字 以上の文字列だった場合には、2-3 文字目から順に左から右へ操作を適用して行くことで、`BAA ... AB` という回文にできる。

また、文字列が A で終わる 3 文字以上の文字列だった場合にも、逆向きに右から左へ操作していけば、`ABB .. BA` を得ることができる。

それ以外の場合、つまり、最初が 'A' で最後が 'B' だった場合には、これらふた文字を変更する手段がないため不可である。

ただし、B から始まり A で終わる文字列のうち、二文字のもの、つまり "BA" だけは回文にできない。

投稿例：33639502[*1]

ARC145B - AB Game

これ、明らかに TLE になるのがわかっていながら、Grundy 数を求めるコードを提出して TLE。

ちゃんと解くのは後日にするとして、Alice は自分のターンでは取れるだけ石をとるべきであるという部分だけ確認しておく。（解説中の「証明」のところ）

Alice のターンにおいて、取れるだけ多くの石を取るべきであることを以下に証明する。

N < A のときには、明らかに Alice の負けである。そこで、以降では $N \geq A$ とする。

$A \leq B$ の場合、Alice は取れるだけ石をとることで、残りの個数を $N \mod A$ 個、つまり A 未満、したがって B 未満にできるので勝つことができる。よって、この場合には Alice は取れるだけ石を取るべきである。

また、$A > B$ の場合、もし Alice が取れるだけ石を取らなかった場合、残りの石の個数が B より多い状況で、Bob の番となり、$A > B$ から先ほどと同様の議論により、Bob は取れるだけ石を取ることで勝つことが出来てしまう。よって、この場合にも、Alice は取れるだけ石を取るべきである。

2022-07-16 (Sat)

典型９０問 074 - ABC String 2

初め単純に貪欲法っぽく書いて TLE。

これ、数字が 3 種類あるけど、3 進数ではないんですよね。でも、先頭に a, b のみがある部分に着目すると 2 進数ではある。なので、2 進数部分は 2 進数として処理して、最初に出現した文字が 'c' の時は素朴に処

[*1] `https://atcoder.jp/contests/arc145/submissions/33639502`

理するように書き換えたら、AC だった。

提出例：33256230[*2]

…なんだけど、最初は 32-bit 整数溢れで WA になっていた。桁数が 60 までなんだから、64-bit 整数使わなきゃってきづこうよ。（解説見て書いたのも同じ間違いをしていて、それで気づいたという）

自分の解法は、解説のものに比べるとやや鈍臭いけど、普通に思いつけるものだと思う。

典型９０問 083 - Colorful Graph

- 少し工夫しつつも普通に解いたやつ： TLE x6[*3]
- Query に登場しない辺を無視することで高速化： TLE x 3[*4]

自分では、ここまでしか思いつかないかなと思って、解説を見てしまった。ここで TLE になるのは、スター型の中心にいるようなノードばかりが呼ばれてしまうケースと考えられるので、一番次数の大きいノードだけ特別扱いしてやればよかった。

提出例：33272979[*5]

実は、解説見てからなかなか AC にならなかったんだけど、引っ掛かってたのはここ：

```
// 次数が最大のノードだけ特別扱いする。
int maxsiz = -1;
int maxi = -1;
for (int i = 0; i < n; i++){
  if (maxsiz < edges[i].size()){
    maxsiz = edges[i].size();
    maxi = i;
  }
}
```

`maxsiz < edges[i].size()` のところで、符号ありなしを比較している旨の警告が出るのを無視していたんだけど、ここで -1 が size_t の最大値にキャストされてしまって、決して true にならないバグになっていた。

いかんね。

2022-07-04 (Mon)

ABC258E - Packing Potatoes

コンテスト本番では、最初 TLE x7 となってしまい、焦って不完全な高速化をやったけど TLE x2 残ってしまいダメだったやつ。

まず、高々長さ N の数列の繰り返しとなることは、見てすぐわかったのだけど（誰でも分かりそうだけ

[*2] https://atcoder.jp/contests/typical90/submissions/33256230
[*3] https://atcoder.jp/contests/typical90/submissions/33269173
[*4] https://atcoder.jp/contests/typical90/submissions/33270105
[*5] https://atcoder.jp/contests/typical90/submissions/33272979

ど)、繰り返しの検出が雑すぎて入力例 2 すら通せず悩むのに時間を幾らか消費してしまったことが 1 つ目の反省点。

次に、X が大きい数になり得るので、尺取り法などを使わないと TLE になるだろうということに気づいていなくて、提出してから焦ってしまったのが 2 つ目の反省点。

提出例は、おととい TLE だったものをもとに修正したものなので、はじめから見通していたら、もうちょっとスッキリ書けたのかなという気はする：32986532[*6]

2022-06-28 (Tue)

典型９０問 049 - Flip digits 2

これは、以前似た問題が解けなかった経験があったのを覚えていたので、解けた：32818990[*7]

典型９０問 054 - Takahashi Number

えるでしゅしゅうだなと思いながら、普通に dijkstra 書いたら TLE。それで★6 なわけがないか。

解説にあるように、仮想的なノードを追加することで、全結合による辺の数を大幅に抑えることができる：32819779[*8]

2022-06-27 (Mon)

ABC257E - Addition and Multiplication 2

貪欲ではうまくいかなさそうだし…と飛ばしてしまったのだけど、もう少し良く考えればよかった。

なるべく大きな数字を得るためには、まず、桁数をより大きくしたい。結果として得られる最大の桁数は、N を C_{min} で割れば得られる。

今度は、より上位により大きな数字（より大きなコストがかかる）を割り当ててやれば良いので、残りの桁のために残しておかなければいけない費用を確保しつつ貪欲に割り当てていけば良い。

投稿例：32798195[*9]

2022-06-04 (Sat)

典型９０問 F - Smallest Subsequence

いっしゅん、部分列なら DP かしらと思ったりしたけど、実例を紙に書いて考えてみたらわかった。[l, r) 区間における最小の文字を探すのに、はじめ segtree を使って、それで AC だったんだけど、解説にあるような LUT を使うやり方も典型っぽいので、そちらでも投稿。

2022-06-02 (Thu)

ABC215F - Dist Max 2

わかんなくて解説みた。解説見ても、いまいちピンとこなくて、書いてみてわかった感。

[*6] https://atcoder.jp/contests/abc258/submissions/32986532
[*7] https://atcoder.jp/contests/typical90/submissions/32818990
[*8] https://atcoder.jp/contests/typical90/submissions/32819779
[*9] https://atcoder.jp/contests/abc257/submissions/32798195

まず、最大や最小の問題はニブタンが使える場合が多いよ、という話。この問題は、これにきづいてからも少々難しかった。x 座標でソートして尺取り法ということなんだが、慣れていないと、さっとは書けないかも。

投稿例：32162635[*10]

ABC216F - Max Sum Counting

こちらは自力で解けた。一度 TLE 出してしまってから、そうかーと思って書き直し：32164544[*11]

2022-05-30 (Mon)

ABC214F - Substrings

「部分列 DP」を知らないとすごく難しく感じるのだけど、知っていると簡単かも。知らなかったので諦めて解説をちょっとみた。部分列 DP の考え方はとてもシンプル。

異なる選び方でも結果的に同じ文字列になる場合は、それらを重複して数えないようにしないといけないわけだが、それがかなり簡単に実現できる。つまり、同じ文字列になる選び方のうち、選ぶ位置の列の辞書順が最も若くなるものだけを選ぶようにすればいい。

実際にどうするかというと、dp で次の文字を選ぶ際に、同じ文字（例えば 'a'）が後ろに複数ある場合には、必ずいちばん前にあるものを選ぶようにすればいい。

今回の問題は、単純な「部分列 DP」ではないが、その処理は次の文字を探し始める位置を変えるだけなので対処は簡単だった。

投稿例：32112003[*12]

2022-05-21 (Sat)

ABC252D - Distinct Trio

解き方が思いつかなくて焦った。順位表を見ると、かなり多くの人が解いているので、ますます焦る。

あまりいい解き方ではないかもしれないけど、

- dp1[i]：i 番目までに 1 文字選ぶ選び方
- dp2[i]：i 番目までに 2 文字選ぶ選び方
- dp3[i]：i 番目までに 3 文字選ぶ選び方

を順に求めていく方法で解いた：31870777[*13]

だけど、解説にあるように、こういう 3 つの数字の組み合わせ (i, j, k) を扱う時に、j を全探索するのは典型的なパターンだったような気がする。気づかなかった：31879128[*14]

※そういえば、「半分全列挙」というテクニックもあったな。本質的には、それと似ているかも。

※列を前から見ていくのと、後ろから見ていくのと、どっちがいいか両方考えてみようというのに加えて、３要素の時には真ん中に着目してみるという選択肢も持っておけということね。

[*10] https://atcoder.jp/contests/abc215/submissions/32162635
[*11] https://atcoder.jp/contests/abc216/submissions/32164544
[*12] https://atcoder.jp/contests/abc214/submissions/32112003
[*13] https://atcoder.jp/contests/abc252/submissions/31870777
[*14] https://atcoder.jp/contests/abc252/submissions/31879128

参考：ここ[*15]の「固定して考える」

もう一つの解説には、DP で解く方法も載っている。本質的には、自分がやったのと似ているんじゃないかなぁと思うんだけど、自分が悩みながら書いたものよりスマートそうなので、そちらも見ておこう。

この問題は、(i, j, k) を動かす解き方の方がスマートだなと思うけど、DP の方は、選ぶ個数が一個増えて (i, j, k, l) の選び方の個数を求めよってされても解けると思うので、そういう意味では悪くはないかなぁ。

ABC252E

解けなかった。くやしい。

で、書いてみたのだが、何回か TLE を出してしまった。ちゃんと正しい dijkstra をかけていなかったようだ。que に経路をプッシュする前に、距離が緩和し、かつ、緩和される時のみプッシュしないといけなかったようだ：31881525[*16]

以前、dijkstra で解けると思って書いたのに TLE だったのこれか。見直してみよう。 ABC246E なんだけど、ちゃんとやってた[*17]。この問題は diijkstra では解けないやつだったのかな

2022-05-17 (Tue)

ABC006D - トランプ挿入ソート

今日も典型問題をやろうと思って、LIS (Longest Increasing Sequence) を。これは蟻本の p.63 にある。すげー見事だ。

で、AtCoder の過去問で LIS をシンプルに扱っているのが ABC006D だったので、これを解いた：31769166[*18]

2022-05-16 (Mon)

ABC014D - 閉路

Euler Tour の典型問題。昨日蟻本や Web ページをいくつか見て Euler Tour と RMC (Range Minimum Query) で LCA を求める方法について学んだばっかりだったので、すぐに書き方はわかったのだけど、ちょこちょこバグって書くのに時間かかってしまった。

これは、さらさらっと書けるようにしておきたい：31751845[*19]

2022-05-15 (Sun)

昨日は D 解けたのに惜しかったなぁという反省はあったものの、レートを少し戻して、もうちょっとで水色復帰かなというところだったのですが…。

連日で挑戦した ARC140 は大惨敗でした！　はは。

[*15] https://algo-logic.info/how-to-think-cp/

[*16] https://atcoder.jp/contests/abc252/submissions/31881525

[*17] 嘘。ループの最初で dist 更新し忘れていた。que に入れる前か、ループの頭どっちかでやればいいと勘違いしていた？

[*18] https://atcoder.jp/contests/abc006/submissions/31769166

[*19] https://atcoder.jp/contests/abc014/submissions/31751845

ARC140A - Right String

いくつものミスがあって、120 分かかっても通せなかった。以下それぞれ反省点などを。大きく 3 つの反省ポイントがあった。

まず、今回も誤読して問題を難しく考えてしまっていた。

「T の先頭の文字を削除し末尾に追加する操作を任意の回数行うことによって作ることのできる文字列の種類数を求めてください。」

を、なぜか、

「T の先頭の『任意の数文字』を削除し末尾に追加する操作を任意の回数行うことによって作ることのできる文字列の種類数を求めてください。」

と誤読してた。こう誤読していても、結局のところ得られる文字列の種類は変化しないので、結局同じ問題を解くことになるのだけど、それに気づくための余分な時間を使ってしまった。

問題よく読もうよ、おれ。

次に、あるシフト幅に対して何文字変える必要があるのかを求めるときに、これは、根本的な間違いをしてしまっていた。

例えば、abcabcabd のような文字列 3 シフトしても変わらないようにすることを考える。これは、最後の一文字を c に変えて abcabcabc にしてやればいいわけだけど、これを計算するのに、3 文字ごとに abc, abc, abd ときった時のどれに揃えてやれば必要な変更回数を最小にできるかと求めてやればいいのかな、と。

これで、問題文にある例は全て解けるのだが、これでは 8WA になってしまう[*20] 。

これは解説にある通り、M シフトの場合には M 箇所それぞれについて、最も出現回数の多いものをターゲットにすればいい。

最後の 3 つ目は、これもなぁ、我ながらへぼくていやんなるんですが、頻度をカウントするための配列の初期化もれ。

最初の初期化もれバージョンはこれ[*21]で、手元では通っていた sample ですら WA していたので、これにはすぐ気づいた。ローカルの配列は 0 初期化されるとは限らないよね、という。

で、何も考えずに（思考停止よくない！）初期化したバージョンがこれ[*22]。

ちゃんとよく考えるべきですよね。M 箇所それぞれ正しくカウントするには、M 回ループの先頭で初期化しなければならない。当たり前ですが！！（自分向けに強めに）

で、やっと通したコードがこれでした：31747334[*23]

やれやれ

2022-05-14 (Sat)

パナソニックグループプログラミングコンテスト 2022 (ABC251) D - At Most 3

これ、100 進 3 桁ともうちょっとだよね、という、解説にある通りの方針には比較的すぐ気づいたのだけど、「や、そんなに簡単なはずないか」という思い込みも手伝って、問題を誤読してしまった。

[*20] https://atcoder.jp/contests/arc140/submissions/31741894
[*21] https://atcoder.jp/contests/arc140/submissions/31742190
[*22] https://atcoder.jp/contests/arc140/submissions/31742489
[*23] https://atcoder.jp/contests/arc140/submissions/31747334

この問題では、W 以下の全ての正整数が「良い整数」であればいいのだけど、これを誤読して、これに加えて「W を超えるものは良い整数にはならない」という条件を勝手に自分で追加してしまって、悩みまくってしまっていた。

問題はよく読もう。

投稿コード：31694717[*24]

2022-05-09 (Mon)

ABC250E - Prefix Equality

昨日のコンテスト中も「なんかハッシュでも解けそうだな」と思ったものの、思っただけだったのと、ハッシュで解くほうが楽そうだったので、まずはそちらで書いてみた。

ここで使えるのは Zobrist Hash というやつ。各要素に対応する乱数値（要素の値から、その値に対応する乱数値へのマップ）を用意しておいて、部分集合に含まれる要素に対応する乱数値の XOR をとるというもの。なんか聞いたことあるのは、将棋やチェスで使われるからかな。

XOR の性質より、計算結果が演算の順序に順序に依存しないので、今回のようなケースに使いやすい。

投稿コード：31572105[*25]

上のコードは無駄なことしてたので書き直し：31589302[*26]

2022-05-08 (Sun)

ABC247E - Max Min

当初思いついた通りのやり方で解けたのだけど、X == Y のケースでバグっていて手間取ってしまった。if 文の羅列を、つい、

```
if (a[i] > x) {
  igtx = i;
} else if (a[i] == x){
  ieqx = i;
} if (a[i] < y) {
  ilty = i;
} if (a[i] == y){
  ieqy = i;
}
```

と書いてしまったためだった。こう書くとバカみたいだけど。

[*24] https://atcoder.jp/contests/abc251/submissions/31694717
[*25] https://atcoder.jp/contests/abc250/submissions/31572105
[*26] https://atcoder.jp/contests/abc250/submissions/31589302

なお、これは、解説、ユーザ解説いずれとも違う（ユーザ解説の方に近いかもしれないけど、解説の方がずっとうまい)。

まー、計算量からも解けるはずと思って、その通り解けたのだからいいのかもしれないけど、解説のやり方も確認しておこう。

ABC250D - 250-like Number

方針はすぐに立って、その方針であっていたのだけど、WA 連発してしまった。

64-bit 整数でも溢れてしまう場合があるケースへの対処がうまくできなかったのが原因だったのだけど、対処したつもりでいたりしたので、的が絞れずに時間と手数を浪費してしまった。

対処法については、解説コードを確認しておこう。

ABC250D 復習

まず、64-bit だと足りないなと気づいた時点で、さっと 128-bit 変数を使うという手があった。中途半端な対策で WA 重ねて混乱するのでなく、最初からこうしておけばよかった的な：31569819[*27]

また、解説にあるように、double で概算しておいて、概算値が long long に収まることを確認してから long long 値を使うなども可能：31569965[*28]

2022-03-27 (Sun)

ABC245D - Polynomial division

コンテストで解けなかったもの。

最初に思いついたのが、指数の小さな B_0 から順に求めていくやり方で、途中で $A_0 = 0$ になるケースが許されているのでマズいなと気づいたものの、方針転換できずにドツボにハマった。

C_{M+N}, A_N がいずれもゼロでないという制約があるので、指数の大きなものから求めていけばよかった。気づきたかった：30510010[*29]

2022-03-05 (Sat)

ABC241F - Skate (diff: 1888 青)

普通に解けたけど、書くのに時間かかりすぎたかな。

29861721[*30]

2022-02-28 (Mon)

ABC239E - Subtree K-th Max

ひさびさの AtCoder 練習。少しだけ考えて解説を見てしまった。なるほど。 K_i の制約が小さいので、これでいけるのね、と、dfs で実装。

[*27] https://atcoder.jp/contests/abc250/submissions/31569819
[*28] https://atcoder.jp/contests/abc250/submissions/31569965
[*29] https://atcoder.jp/contests/abc245/submissions/30510010
[*30] https://atcoder.jp/contests/abc241/submissions/29861721

・・・なのだけど、dfs でミスってデバッグに手間取ってしまった。dfs ルーチンの中で、すでに到達済ノードだった場合に何もしない分岐は最初から入れてあったのだけど、呼び出しがわでもチェックして、先のノードが巡回済だったときには配列の追加をしないようにすべきだった。

dfs の書き方が自分の中で定まってなかったんだけど、読み出しもとの方で一つ先が巡回済みかどうかチェックするようにした方が安全なのかもしれないな。

ところで、ユーザ解説を見ると「オイラーツアー」なる用語が見える。

オイラーツアーでは、部分木クエリ、パスクエリ、LCA などができるらしい。あとで勉強しよう。

提出コード： 29774213[*31]

[*31] https://atcoder.jp/contests/abc239/submissions/29774213