ABC025 解説

問題A – 25個の文字列

問題概要

- 長さ 5 の文字列 S と整数 N が与えられます。
- S に含まれる文字を 2 回選び、長さ 2 の文字列を作成します。
- 考えられうる 25 通りの文字列のうち、辞書順で N 番目の文字列を出力してください。

- $1 \leqq N \leqq 25$
- S には 5 種類のアルファベットが使用されており、それらは昇順に並んでいる。

解法

- 25 通りすべての文字列を生成し、ソートをすることで解くことができます。

- ですが、実際にはソートしなくても答えを計算することができます。

- 答えの文字列を X とし、$N-1$ を 5 で割った商と余りをそれぞれ A と B としたとき、X の 1 文字目は S の $A+1$ 文字目、X の 2 文字目は S の $B+1$ 文字目となります (S 内の文字は昇順に並んでいるため)。

問題B – 双子とスイカ割り

問題概要

- N 個の移動についての情報(東に 3 メートルとか西に 7 メートルとか)が与えられます。
- N 個の移動後にどこにいるのかを計算してください。
- ただし、移動距離の指定が A メートルより少ないなら A メートルの移動に、B メートルより多いなら B メートルの移動に変更されます。

- $1 \leqq N \leqq 100$
- $1 \leqq A \leqq B \leqq 100$
- $1 \leqq$ (指定距離) $\leqq 100$

解法

- 上から順にシミュレートします。
- 西方向へ X メートル移動する処理を、東方向へ $-X$ メートル移動するとして、現在位置も初期位置から東に Y メートル(初期位置よりも西側なら Y は負の値になる)としておくことで、移動の処理が簡単になります。

問題C – 双子と○×ゲーム

問題概要

- 3×3 のマス目に交互に〇と×を書いていきます。
- 全て埋まった後に、上下あるいは左右で隣接する 2 マスそれぞれについて、同じ文字が書かれているか異なる文字が書かれているかを判定し、同じなら直大くんに、異なるなら直子さんに得点が入ります。
- 両者が最善を尽くしたときのそれぞれの得点を計算してください。

- $1 \leqq b_{i,j} \leqq 100$
- $1 \leqq c_{i,j} \leqq 100$

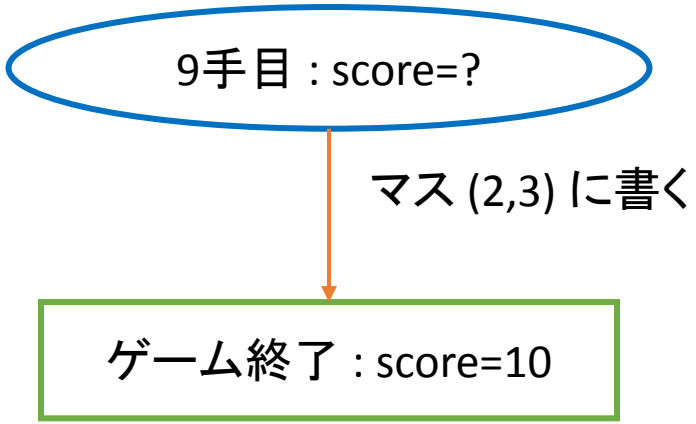
方針

- 1 手目からいきなり 9 手目後の状態を考えるのは大変です。
- 逆に 9 手目から考えてみることにします。

- また、この問題の場合、直大くんの得点と直子さんの得点の合計値は一定なので、直大くんは score=(直大くんの得点)-(直子さんの得点)の最大化を、直子さんは score=(直大くんの得点)-(直子さんの得点)の最小化を目標として行動するものとすれば、扱いが簡単になります。

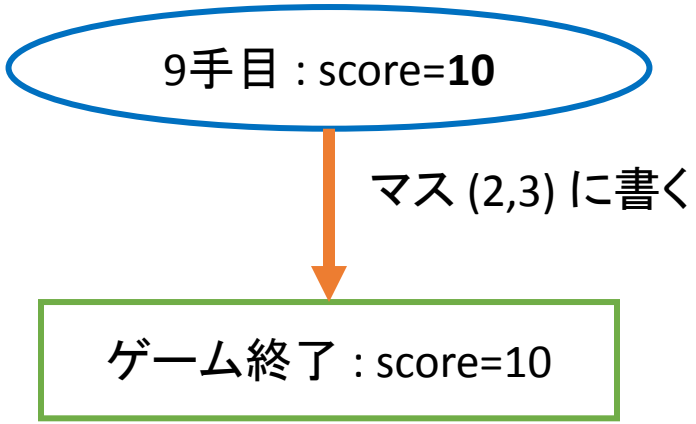
9 手目の状況

- 9 手目において、書くことのできる場所は 1 箇所しか無いので、書く場所は一意に決まります。



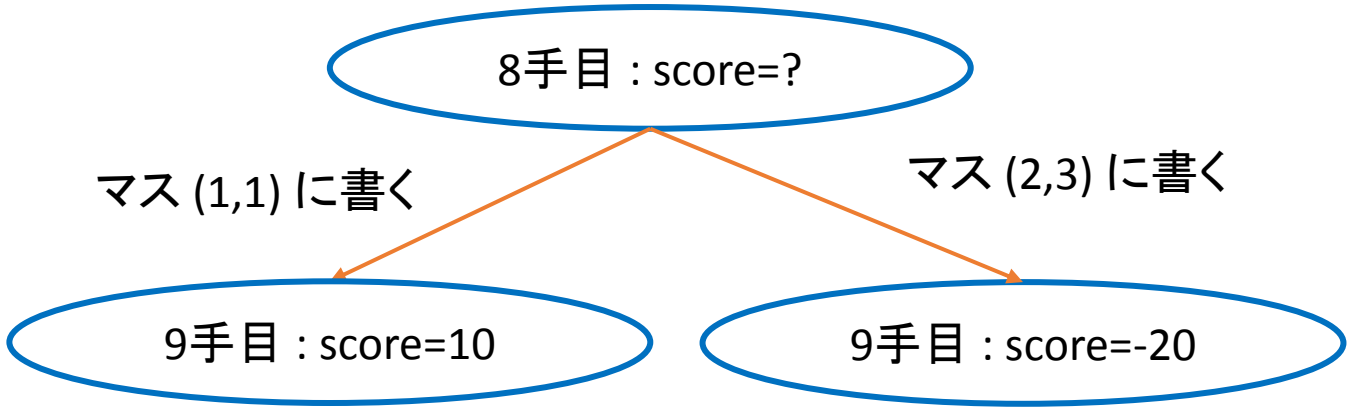
9 手目の状況(直大くんの手番)

- 9 手目において、書くことのできる場所は 1 箇所しか無いので、書く場所は一意に決まります。
- そのため、9 手目の得点は、ゲーム終了時の得点と同じになります。



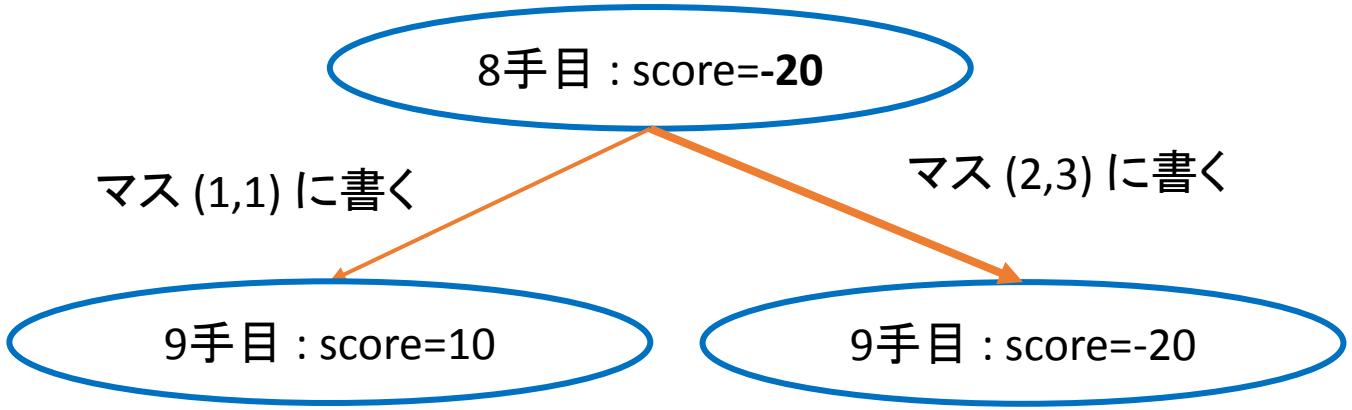
8 手目の状況(直子さんの手番)

- 8 手目において、書くことのできる場所は 2 箇所あります。
- それぞれのマスについて、そこに×を置いた際に、9 手目以降がどうなるかは、先ほどの情報より分かります。



8 手目の状況(直子さんの手番)

- 8 手目において、書くことのできる場所は 2 箇所あります。
- それぞれのマスについて、そこに × を置いた際に、9 手目以降がどうなるかは、先ほどの情報より分かります。
- 直子さんは score を小さくしたいので、遷移可能な点のうち score が最小な点に遷移すれば良いことになります。その値が 8 手目の score となります。



7 手目よりも前について

- 8 手目の score が分かるので、7 手目、6 手目、…とわかっていきます。
- 最終的に 1 手目において直大くんが置いたときの score が、両者が最善を尽くしたときの score となります。

- まとめると、以下の再帰関数を実装すると解けます。
    1. 現時点で置き場所がないなら、score を直接計算し返します。
    2. すべての置き場所に対し、一旦置いてみることにします。
    3. 置いた後の score をこの再帰関数を呼び出すことで計算します。
    4. 直大くんの手番ならそれらの最大値を、直子さんの手番なら
       それらの最小値を再帰関数の返り値とします。

備考

- この探索は、ゲーム木の探索です。
- 計算量は $O(N!)$ (N は手数)となります。
- 手数が少ないので、特に高速化しなくても計算量的には解けます。

- メモ化再帰をすることで $O(2^N)$ にできます。
- ゲーム木の探索アルゴリズムなので、αβ枝刈りなどもできます。

問題D – 25個の整数

問題概要

- 1 から 25 までの整数を 5 × 5 の盤面に配置します。
- いくつかの整数はどこに置くか決まっています。
- 縦または横に連続する 3 整数をどのようにとってもそれらが昇順または降順になっていない置き方は全部で何通りあるか計算してください。

- 少なくとも 5 つの整数は置き方が決まっている。

部分点解法 (30点)

- すべての置き方を試します。
- 置き方は $N!$ 通り (N は空きマスの個数)あり、それぞれ $O(N^2)$ で条件を満たすか判定できるので、計算量は $O(N^2 * N!)$ となります。
- 空きマスが少ないデータセット 1 なら高速で解けますが、空きマスが多めのデータセット 2 だとかなりの時間がかかってしまいます。

考察

- 1 から順に置いていくことにします。
- この場合、どのような条件が満たされたら良くて、どのような条件が満たされたらダメなのかを考えてみます。

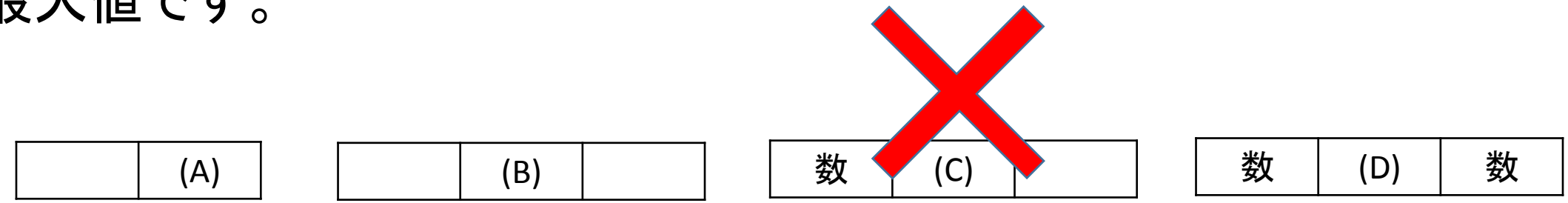
考察

- 今新たに数を置くときに、上下または左右で隣接するマスについて考えてみます (ここでは例として左右を考えます、上下も同様に処理できます)。以下の 4 つの場合が考えられます。

- (A) : 左右のうち一方が盤面の外 (端のケース)
- (B) : 左右のうち両方が空きマス。
- (C) : 左右のうち一方が埋まっている。
- (D) : 左右のうち両方が埋まっている。

考察

- (A)のケースは、条件に違反することはありません。
- (B)のケースは、今置く数を置いた時点では矛盾しません。真ん中が最小値です。
- (C)のケースは、後で空いているマスにより大きい数を置くことになるので、この時点で矛盾していることが分かります。
- (D)のケースは今置く数を置いた時点では矛盾しません。真ん中が最大値です。



考察

- 先ほどの (C) を検出して除外するアルゴリズムだけで、すべての正しい置き方を列挙することができるのでしょうか?

考察

- 先ほどの (C) を検出して除外するアルゴリズムだけで、すべての正しい置き方を列挙することができるのでしょうか?

- →これできちんとチェックできます。なぜなら、連続する 3 マスについて、それらの大小関係はちょうど 2 個目が置かれたときに定まり、かつダメなケースでは必ず (C) を経由することになるからです。
- 一方で(C)さえ経由しなければ最後まで置けた配置はいずれも条件を満たします。

満点解法 (30+70点)

- もちろん先ほどのチェックを導入してもただすべての置き方をそのまま試すだけでは、結局 $O(N!)$ 通り見ることになります。
- 重要なこととして、先ほどのチェックに用いたのは、今現在置く数との大小関係のみなので、既に置いた数同士の大小関係は覚えなくても良いです。
- そのため、既に置いた場所の集合が同じ状態同士を一緒くたにして計算することができます。

満点解法 (30+70点)

- ここで、bitDP と呼ばれるアルゴリズムを使用することができます。
- dp[i] = 今現在置かれている数字の配置が整数 i で表されるとき、現時点で矛盾なく置かれている配置の総数。
- i は 25 ビットの非負整数で、盤面をビットで表します (例えば 10 マス目が埋まっているなら第 10 ビットが 1 でそうでないなら第 10 ビットが 0)。
- このとき、$dp[i]=\sum_{j=1}^{25} dp[i-2^{j-1}] * f(i-2^{j-1}, j)$ となります。ただし、ここで $f(i, j)$ は、i の第 j ビットが 0 であり、状態 i から新たに数を j マス目においても矛盾しない場合に 1 を、それ以外の場合に 0 を返す関数とします。また、dp[0]=1 とします。

満点解法 (30+70点)

- 先ほどの bitDP は $O(2^N)$ の計算量で動作します。
- $N \leqq 20$ なので、データセット 2 でも速く動作します。