審判長講評

審判長　鵜川 始陽

最初に統計データを示すことにしよう。正解数ごとのチーム数を表 1 に示し、各問題の正答チーム数、誤答チーム数、提出数を表 2 に示す。

表 1: 正解数ごとのチーム数

正解数	11	10	9	8	7	6	5	4	3	2	1	0
チーム数	0	0	2	0	1	8	7	11	11	2	1	0
チーム数累計	0	0	2	2	3	11	18	29	40	42	43	43

表 2: 問題ごとの正答チーム数、誤答チーム数、提出数

問題	A	B	C	D	E	F	G	H	I	J	K
正答チーム数	43	42	0	23	19	19	37	2	1	1	3
誤答チーム数	0	1	4	10	5	3	2	1	2	2	4
提出数	63	79	6	81	41	42	69	3	3	3	22

また、審判団が想定していた難易度の順に並べた問題番号と正答数を表 3 に示す。

表 3: 想定難易度順の問題番号と正答数

問題	A	B	G	D	H	J	E	C	K	F	I
正答数	43	42	37	23	2	1	19	0	3	19	1

今年は 3 年ぶりにオンサイトで開催することができた。審判団も前々日から会場に詰めて問題の仕上げを行ったが、会場の準備や選手が集まってくる様子を目にし、コンテストが始まるというのを実感できた。

審判団では最初に、前回（2022 年 3 月開催の ICPC 2021）よりも難易度を下げるという方針を確認した。前回はオンライン向けに難しい問題セットにしたが、今年度はオンサイト開催になるためである。結果としては、オンライン開催だった前回よりも多くの問題が解かれ、難易度の設定は適切だったように思える。

しかし、表 3 に現れているように、必ずしも想定難易度の易しい順に解かれている訳ではない。特に、難しい問題として出題した K, F, I の 3 問のうち K と F は比較的解かれている一方、中盤問題として出題した H, J, C は正解が少ない。これらの問題は提出数が少ないので、あまり手をつけられていなかった可能性がある。問題 A, B 以外の想定難易度はコンテスト中は選手に分からず、他のチームの正解状況だけが分かる。そのため、難しい問題であっても、正解したチームがあるとそれに引きずられて手をつけてしまう傾向がある。今回もそのようなことが起き、その結果、このような「荒れた」コンテストになった。

今回は、日本の地区大会では初めてインタラクティブ問題を出題した。どの難易度のインタラクティブ問題を出題するかについては議論があった。全チームにチャレンジしてもらいたいという思いがある一方、インタラクティブ問題には通常問題とは違う入出力の取り扱いが要求され、問題の本質以外にも難しさがある。最終的には、易しい問題の枠で出題し、事前にインタラクティブ問題で

気をつける点を周知したり、前日のリハーサルにもインタラクティブ問題を含めるなどで、選手が本質でないところでつまずくのを防いだ。大会の本番では大きなトラブルもなく、ほぼ全てのチームが正解にたどりついた。

以下では各問題について、解法を中心に簡単に解説する。

問題 A: Hasty Santa Claus

n 個の整数の区間が与えられ、それぞれの区間から整数を一つずつ選ぶ問題である。ただし、同じ整数は高々k 個しか選んではならない。問題文は、留守を狙ってプレゼントを配るサンタクロースのスケジューリングとして仕立てた。区間は配る先の n 軒の家が留守にしている期間で、選択するのはプレゼントを配る日である。サンタクロースは 1 日に高々k 軒しかプレゼントを配ることができない。

この問題は、スケジューリングでは典型的な貪欲アルゴリズムの解法を想定していた。具体的には、区間の終点が小さい家から順にプレゼントを配る日を割り当てる。このとき、区間の条件と、1 日に配れる数の条件を満たす最も早い日を割り当てる。

問題 B: Interactive Number Guessing

日本の地区大会では初めてとなるインタラクティブ問題である。通常の問題では、一度に全ての入力を受けとり、それに対する正解を計算して出力するプログラムを作る。一方、インタラクティブ問題では、競技者が作る解答プログラムは審判プログラムに対して質問をして、その返答から次の質問を決めたり、正解を求めたりする。

この問題では、解答プログラムが秘密の数 n を当てる。解答プログラムは、審判プログラムに対して、0 以上の数 a を送る。それに対して審判プログラムは $n + a$ を 10 進数で表記したときの、各桁の数字の総和を返答する。例えば $n = 75$ のとき、解答プログラムが $a = 3$ の質問を送ると、審判プログラムは $75 + 3 = 78$ の各桁を合計した 15 を返答する。

$0 \leq n, a \leq 10^{18} - 1$ なので、$a = 0$ の質問をして $n = 0$ の場合を除外したうえで、$a = 10^{18} - 1$ から順に a を 1 ずつ小さくしながら質問をすれば、いつかは正解を見つけることができる。$a + n = 10^{18} = 100 \cdots 0$ のとき初めて返答が 1 となるからである。しかし、質問の回数が 75 回以下と制限されているので、もっと効率のよい質問の戦略を考えなければならない。

想定解法では、返答から $a + n$ の計算で繰り上がりが起こったかどうかを判定できることを利用する。もし繰り上がりが起こっていなければ、$a + n$ の各桁の和は、a と n で独立に各桁の和を計算して、それを足した値になる。しかし、もし繰り上がりが起こっていれば、$a + n$ の各桁の和は独立に計算して足した値より小さくなる。これを利用すると、a に 1 桁だけ 0 でない数 $b \times 10^c$ を指定し、10^c の位が $10 - b$ 以上かどうかを判定できる。最初に $a = 0$ の質問をした後、1 桁につき 4 回の質問を使って二分探索すればその桁の数を確定できるので、18 桁であれば $1 + 4 \times 18 = 73$ 回の質問で n の値を求めることができる。

問題 C: Secure the Top Secret

平面上のグリッドの格子点に柱を立て、隣接する柱の間のいくつかを壁や開閉可能なシャッターで仕切ったような長方形の建物を考える。4 本の柱で囲まれたグリッドの 1 マスを単にマスと呼ぶことにしよう。この問題では、入口のマス (S) から秘密があるマス (T) への経路を確保しつつ、S 以外に外から侵入できるマス (U) から T への経路ではどの経路をとっても U と T が 2 枚以上のシャッター

で隔てられるように、閉じるシャッターを選択する。ただし、S, U だけでなく、T も建物の外周部にある。また、建物の外周部には全て壁が設置されている。

一見してグラフ問題だが、この問題で考えるべきグラフは、マスを頂点とするグラフ（これを G とする）ではない。格子点を頂点とし、壁や閉じたシャッターを辺とするグラフ G' である。ただし、G' では建物の外周の壁のうち、S, T, U のマスの壁は取り除く。これにより、建物の外周部は三つに分割される。これらを外周区間 S-T のように呼ぶことにしよう。

問題文では G に対する条件が与えられているので、これを G' に対する条件に言い換える。例えば、G で「U から T に経路がない」は、G' では「外周区間 U-T から外周区間 S-T か外周区間 S-U に経路がある」と言い換えられる。問題の条件はもう少し複雑で、G で「U から T への全ての経路が 2 枚以上のシャッターで隔てられる」（条件 1）である。これは G' では「外周区間 U-T から外周区間 S-T か外周区間 S-U に二つ以上経路があり、それらは同じシャッター辺を使わない」（条件 1′）と言い換えられる。また、もう一つの条件である G で「S から T に経路がある」（条件 2）は、G' では「外周区間 S-T から他の外周区間に経路がない」（条件 2′）と言い換えられる。条件 1′ は条件 2′ を加味すると「外周区間 U-T から外周区間 S-U に二つ以上経路があり、それらは同じシャッター辺を使わない」（条件 1″）と簡単にできる。

このように言い換えて、最小費用流の問題に帰着する。つまり、シャッターを全て閉じたときのグラフ G' を考える。このグラフ上で、外周区間 S-T と、そこから壁 (シャッターではない) で連結している壁辺には容量 0 を、残りの壁辺には容量 ∞、コスト 0 を、シャッター辺には容量 1、コスト 1 を設定し、外周区間 S-U から外周区間 U-T への流量 2 の最小費用流を求める。

問題 D: Move One Coin

ある形に並べられたコインのうち 1 枚だけを動かして、別の形を作る。そのとき動かすべきコインを求める問題である。コインを動かした後の形は、90 度の整数倍回転させた形で与えられるが、4 通り試せばよいだけなので、ここでは回転はしていないとして説明する。

コインを動かす前と後の形を重ねることができれば、位置の違うコインは簡単に見つかる。しかし、重ねるためには、コインを動かす前後の形で対応する、基準となる点を決める必要がある。もし、この問題が、コインを動かさずに合同な図形を重ねるだけの問題であれば、コインの位置が辞書順で先頭である（x 座標が最小なコインの中で y 座標が最小である）コインを基準にできる。つまり、二つの図形の辞書順先頭のコイン同士が同じ位置になるように重ねれば、二つの図形はぴたりと重なる。しかし、この問題ではそのコインを動かす可能性があるので、単純に辞書順先頭のコインを基準にすることはできない。

この問題を解く上で鍵となるのは、動かすコインが一つだけであるという条件である。もし辞書順先頭のコインを動かす場合、辞書順で 2 番目のコインは動かさない。コインを動かした後の形についても、辞書順先頭のコインはもともとは別の位置にあった可能性があるが、その場合は辞書順で 2 番目のコインは動かしていないことになる。したがって、コインを動かす前後の形のそれぞれで、辞書順で先頭のコインを基準にする場合と 2 番目を基準にする場合の、合計 4 通りを試せば、その中に正解がある。正解のときはコインを 1 枚動かすだけで与えられた形が得られる。

基準の選び方は、この他にも様々な方法が考えられる。例えば、コインが十分に多ければ、全てのコインの重心を基準としてもよい。コインを置ける範囲は決まっているので、コインが十分に多いと、その範囲の中でコインが 1 個動いたぐらいでは重心の位置の座標が 1/2 も変化しなくなる。一方、コインは整数座標にあるので、重心同士を重ねた後、コインを最も近い整数座標にずらせば、ぴたりと重なる。

問題 E: Incredibly Cute Penguin Chicks

文字列を ICPC-ish という条件を満たす文字列に分割するとき、何通りの分割方法があるかを求める問題である。

競技プログラミングに慣れていれば、動的計画法を使うのだろうというのは簡単に察しが付くだろう。入力 S の先頭 t 文字だけを取り出した文字列に対する分割方法の数 $d(t)$ を順に計算するのである。$d(t)$ の計算では、$t' < t$ である各 t' について、入力の $t'+1$ 文字目から t 文字目までを切り取った文字列（以降、$S[t'+1, t]$ と書く）が ICPC-ish かどうか判定する。ICPC-ish であれば、$S[t'+1, t]$ を一つの ICPC-ish 文字列とするような分割の方法が $d(t')$ 通りある。したがって、そのような t' 全てについての $d(t')$ を合計すれば、$d(t)$ が計算できる。この方法では $d(t)$ の計算に入力文字列長 n に対して $O(n)$ 時間かかり、アルゴリズム全体では $O(n^2)$ 時間かかる。しかし、$n \leq 10^6$ なので、これでは残念ながら時間内に正解を求めることはできない。

想定解法では、t' をグループに分類して管理し、$S[t'+1, t]$ が ICPC-ish になるような $d(t')$ を高速に探せるようにする。`C` だけが多い ICPC-ish 文字列を C-ICPC-ish 文字列と呼ぼう。C-ICPC-ish 文字列は、`I` と `P` を同数含む。そこで、ある t において t' を、$S[t'+1, t]$ に含まれる `I` の数から `P` の数を引いた数 k で分類し、集合 C_k を作る。$t' \in C_0$ である $S[t'+1, t]$ が C-ICPC-ish 文字列の候補であり、そのうち `C` が他の文字より多く出現する文字列が C-ICPC-ish である。さらに、C_k の要素を $S[t'+1, t]$ に含まれる `C` の数から `P` の数を引いた数の順に並べておけば、C-ICPC-ish 文字列を作る t' は連続する。

集合 C_k の添字は、次の文字を読んだときに一斉にずらす。つまり、次の文字が `I` であれば、C_0 を C_1 に、C_1 を C_2 にというように 1 ずつ増やす。実装では全ての集合の添字をずらすのではなく、集合の添字を $k+\alpha$ としておき、α を増減させることで定数時間で全ての集合の添字をずらす。こうして新しく C_0 になった集合の要素である t' から作った文字列 $S[t'+1, t+1]$ が新しい C-ICPC-ish 文字列の候補である。順序付き集合 C_k の要素を t' の代わりに $d(t')$ にし、それを Fenwick 木を用いて実装すれば、$O(\log n)$ 時間で C-ICPC-ish を作る t' に対する $d(t')$ の総和が求められる。

以上を、I-ICPC-ish 文字列や P-ICPC-ish 文字列についても計算するのだが、この計算は二次元座標系を使って、次のように簡潔に整理できる。文字列を先頭から 1 文字ずつ読み、それに従って二次元座標の格子点の上を移動する点 A を考える。この点は `C` を読むと $(1, 0)$、`I` を読むと $(0, 1)$、`P` を読むと $(-1, -1)$ だけ移動する。そして、t 文字目まで読んだとき、その座標に $d(t)$ を記録する。同じ点を複数回訪れることもあるが、その時は訪れる度に $d(t)$ を加算する。

ここでは、$S[t'+1, t]$ に含まれる `I` と `P` の数の差は、t' 文字目まで読んだときの点 A の Y 座標と t 文字目まで読んだときの Y 座標の差になる。これが（座標を複数回訪れたときの $d(t)$ は個別に記録せず総和にした）順序付き集合 C_k の添字であり、C_k の要素は X 軸に平行な直線上に並ぶ。また、点 A の X 座標が `C` の数から `P` の数を引いた数になる。I-ICPC-ish 文字列については、Y 軸方向について同様に考えればよい。また、P-ICPC-ish 文字列については、$(-1, -1)$ 方向について考えればよい。ここでは、Y 軸（または X 軸）方向にいくらずれているかが `I` と `C` の文字数の差を表す。

問題 F: Make a Loop

プラスチックのレール部品を連結して環状の線路を作り、その上に列車の模型を走らせるおもちゃで遊んだ人は多いだろう。この問題では、様々な回転半径を持つ 90 度分の円弧の形状をしたカーブのレール部品だけで線路を作る。与えられたレール部品を全て使って環状の線路を作ることができるかどうかを判定するのが、この問題である。

環状の線路を作るには、一周したときに最後のレール部品の終点が最初のレール部品の始点と同じ座標になければならない。これを「位置条件」と呼ぶことにする。いま、1 から n の番号を振った n 個のレール部品を使って環状の線路ができているとしよう。そのうちのレール部品を一つ選び、その半径を r としたとき、一方の端（始点）が原点、他方の端（終点）が (r, r) となるような座標を考える。また、他のレール部品にも、このレール部品の始点から終点に向かうように環状の線路を一周する方向で向きを付ける。位置条件は、各レール部品の始点と終点の座標の差を求めて合計したときに、X 座標も Y 座標も 0 になると言い換えられる。どのレール部品も 90 度の円弧なので、レール部品 i の半径を r_i としたとき、その始点と終点の座標の差の絶対値は X 座標、Y 座標とも r_i となる。符号はそのレール部品がどの向きに置かれているかによる。始点に対して終点が第一象限に来るように置かれているレールの番号の集合を $S_{+,+}$ としよう。同様に、第二象限、第三象限、第四象限をそれぞれ $S_{-,+}$, $S_{-,-}$, $S_{+,-}$ としよう。これらを使って、位置条件を次のように書ける。

$$\sum_{i \in S_{+,+} \cup S_{+,-}} r_i - \sum_{i \in S_{-,+} \cup S_{-,-}} r_i = 0$$
$$\sum_{i \in S_{+,+} \cup S_{-,+}} r_i - \sum_{i \in S_{+,-} \cup S_{-,-}} r_i = 0$$

さらに、最後のレール部品の終点と最初のレール部品の始点を原点で連結するためには、それらが同じ向きに向いていなければならない。これを「向き条件」と呼ぶことにする。原点での線路の向きを 0 度として、反時計回りに角度を考えよう。$S_{+,+}$ のレール部品には 0 度の向きの線路を 90 度にする置き方（反時計回り置き）か、90 度の線路を 0 度にする置き方（時計回り置き）がある。同様に $S_{-,+}$, $S_{-,-}$, $S_{+,-}$ でも反時計回り置きと時計回り置きの二通りの置き方がある。向き条件を満たすには、それぞれの集合の反時計回り置きをしたレール部品の数から時計回り置きをしたレール部品の数を引いた値が等しい必要がある。それには、次の式の条件を満たす必要がある。

$$|S_{+,+}| \equiv |S_{+,-}| \equiv |S_{-,-}| \equiv |S_{-,+}| \pmod 2$$

これに加えて、環状の線路を作るには、いずれの集合も空であってはならない。

$$S_{+,+}, S_{+,-}, S_{-,-}, S_{-,+} \neq \emptyset$$

実は、以上の条件が線路が環状になるための必要十分条件である。与えられたレール部品を上の条件を満たすように四つの集合 $S_{x,y}$ $(x, y \in \{+, -\})$ に分割できるなら、環状の線路を作ることができる。このような四分割が存在することは、

$$\sum_{i \in S_{+,*}} r_i - \sum_{i \in S_{-,*}} r_i = 0$$
$$\sum_{i \in S_{*,+}} r_i - \sum_{i \in S_{*,-}} r_i = 0$$
$$|S_{+,*}| \equiv |S_{-,*}| \equiv |S_{*,+}| \equiv |S_{*,-}| = 0 \pmod 2$$
$$S_{+,*}, S_{-,*}, S_{*,+}, S_{*,-} \neq \emptyset$$

を満たすような二通りの分割、つまり、$S_{+,*}$ と $S_{-,*}$ への分割と、$S_{*,+}$ と $S_{*,-}$ への分割が存在することと同値である。なぜなら、例えば $S_{+,+} = S_{+,*} \cap S_{*,+}$ とすれば、元の条件を満たす $S_{+,+}$ が得られるからである。二通りの分割は、それぞれ、和が等しく、要素数が偶数になるような分割である。これは部分和問題なので動的計画法で求められる。

問題 G: Remodeling the Dungeon

迷路の入口から出口までの最短経路が最も長くなるように壁を一枚壊して一枚追加する問題である。うまく壁を一つ壊すと、入口から出口までの経路が二つできる。元の最短経路と、その途中のマス A で脇道に逸れ、壊した壁を通過して、マス B で元の経路に合流して出口に至る「寄り道経路」である。その後、元の最短経路上の AB 間に壁を追加すれば、この寄り道経路が最短経路になる。したがって、寄り道経路が最長となるように壊す壁を決めればよい。ただし、A, B が同じマスになるような壁、つまり、壁の両側が元の最短経路の同じマスから分岐した先にあるような壁は、壊しても寄り道経路が作れない。

ある壁を壊したときの寄り道経路の長さは、入口から壁の入口側（壁に隣接するマスのうち元の最短経路からマス A で分岐した先の側）までの長さと、出口から壁の出口側までの長さの和に、壊した壁を通過するコスト 1 を加えることで計算できる。壁に隣接する二つのマス C, D のうち、どちらが入口側かは分からないが、入口から C までの長さと D から出口までの長さの和（α としよう）か、入口から D までの長さと C から出口までの長さの和（β としよう）のどちらか短い方が寄り道経路の長さになる。もし、$\alpha = \beta$ だったときは、寄り道経路が作れないような壁である。

したがって、全ての壁について α と β を計算し、$\alpha \neq \beta$ の中で $\min(\alpha, \beta)$ が最大となる壁を探せばよい。なお、壁ごとに両側のマスの入口や出口からの距離を毎回計算すると、計算が間に合わない。想定解法では、あらかじめ入口と出口それぞれから迷路全体を探索して、各マスの入口からの距離と出口からの距離を計算しておく。

問題 H: Cake Decoration

整数 L, R, X に対して次の条件を満たす異なる整数の組 (d, c, r, b) の数を求める問題である。

1. $L \leq d + c < R$
2. $dcrb \leq X$
3. $(d+1)crb > X$ かつ $d(c+1)rb > X$ かつ $dc(r+1)b > X$ かつ $dcr(b+1) > X$

整数を小さい順に並べて、A, B, C, D と名前を付け直そう。そうすると 2 以降の条件は

$2'$. $ABCD \leq X$

$3'$. $(A+1)BCD > X$ かつ $A(B+1)CD > X$ かつ $AB(C+1)D > X$ かつ $ABC(D+1) > X$

と書ける。条件 1 は d と c が A から D のどれに割り当てられているか分からないので、12 通りそれぞれの場合を分けて場合の数を計算する。ただし、d と c を逆に割り当てても場合の数は同じなので、6 通り計算して 2 倍すればよい。また、r と b の割り当ても 2 通りあるので、求めた場合の数を 4 倍して合計すると、正解が得られる。

それぞれの場合の数を求めるために、条件の式を変形しよう。$A < B < C < D$ なので、条件 $3'$ の左辺の中では $ABC(D+1)$ が最小になる。したがって、条件 $2'$ と $3'$ は次のようにまとめられる。

$$ABCD \leq X < ABC(D+1)$$

A, B, C, D は整数なので、この式はさらに

$$D = \left\lfloor \frac{X}{ABC} \right\rfloor$$

と言い換えられる。つまり、A, B, C が決まると D は一つに決まる。ただし、そのような A, B, C, D は条件 1 を満たさないこともある。

想定解法では、A, B を全通り列挙し、それぞれについて D が存在する C の範囲を求める。C の範囲の求め方は d と c を A から D のどれに割り当てるかによって異なる。ここでは、$d = A$, $c = C$ の場合だけ説明する。このとき $L \leq A + C < R$ を満たす必要がある。C の最小値を与える条件は $L \leq A + C$ と $B < C$ である。A, B は全通り列挙するので、具体的に分かっている。したがって、二つの条件を $C \geq \max(B + 1, L - A)$ とまとめると、右辺は計算できる。次に、C の最大値を与える条件は $C < D$ である。C は $D = C + 1$ のとき最大になるので、C は $C(C + 1) \leq \frac{X}{AB}$ を満たす範囲で大きくなれる。そのような C の最大値は二分探索で求められる。これ以外の 5 通りの d と c の割り当てでも同様に考えて各 A, B について C の範囲を求め、それを合計する。

問題 I: Quiz Contest

極めて効率的なアルゴリズムを構築する必要がある、非常に難しい問題である。解法は以下のような $O(m \log n \log m)$ 時間の分割統治のアルゴリズムを想定していた。

参加者全体の集合を $U = \{1, \ldots, n\}$ とする。U の任意の部分集合 S について、S に属する参加者が正解する $a(S) = \sum_{i \in S} a_i$ 問の並べ方 $a(S)!$ 通りのうち、S に属する参加者のみで大会を行った場合に i 問目で勝負の決着がつくような並べ方の数を $f(S, i)$ $(i = 1, \ldots, a(S))$ と書こう。すると、二つの重なりのない参加者の集合 S と T について、$f(S, \cdot)$ と $f(T, \cdot)$ の値の一覧が既知ならば、$f(S \cup T, \cdot)$ の値の一覧が次の式で計算できる。

$$f(S \cup T, i) = \binom{a(S \cup T)}{a(S)}^{-1} \sum_{j+k=i} \left(f(S, j) \left(\sum_{k'=k+1}^{a(T)} f(T, k') \right) \binom{j-1+k}{k} + f(T, k) \left(\sum_{j'=j+1}^{a(S)} f(S, j') \right) \binom{j+k-1}{j} \right) \binom{a(S \cup T) - i}{a(T) - k}$$

これは、累積和などを使用して整理すると、シンプルな畳み込みに帰着でき、FFT アルゴリズムを用いると $O(a(S \cup T) \log a(S \cup T))$ 時間で計算できる。したがって、n 個の単元集合 $\{1\}, \ldots, \{n\}$ から始め、$a(S)$ の値が最小の 2 集合の併合を繰り返すことで、$O(m \log n \log m)$ 時間で $f(U, i)$ を計算できる。なお、$f(\{i\}, j)$ は $i = j$ のとき 1、それ以外は 0 になる。

次に、U の任意の部分集合 S について、$a(U)$ 問の並べ方の中で『S に属する参加者が正解する問題のみを取り出して部分列を構成し、その参加者のみで大会を行った場合に i 問目で決着がつく』ような並べ方のうち、全員で大会を行った場合にも S に属する参加者が勝利するような並べ方の数を $g(S, i)$ $(i = 1, \ldots, a(S))$ と書こう。すると、二つの重なりのない参加者の集合 S と T について、$f(T, \cdot)$ と $g(S \cup T, \cdot)$ が既知なら、$g(S, \cdot)$ が次の式で計算できる。

$$g(S, i) = \binom{a(S \cup T)}{a(S)}^{-1} \sum_{j=i}^{i+a(T)} \binom{j-1}{i-1} \binom{a(S \cup T) - j}{a(S) - i} \left(\sum_{k=j-i+1}^{a(T)} f(T, k) \right) g(S \cup T, j)$$

これも FFT アルゴリズムを用いることで $O(a(S \cup T) \log a(S \cup T))$ 時間で計算できる。したがって、$g(U, i) = 1$ $(i = 1, \ldots, a(U))$ から始めて、上とは逆の過程で分割を繰り返すことで、$O(m \log n \log m)$ 時間で $g(\{j\}, \cdot)$ $(j = 1, \ldots, n)$ を計算できる。

これを用いて、参加者 i が勝つ並べ方は $m! \cdot g(\{i\}, b_i)$ 通りと計算できる。

問題 J: Traveling Salesperson in an Island

多角形とその周上の点がいくつか与えられ、それらの点を全て通り、多角形の外に出ない最短の巡回路を求める問題である。

この問題では、与えられた点を時計回り（または反時計回りに）に回る巡回路が最短となる。もし時計回りでない巡回路が最短であるとすれば、その巡回路では経路が交差する。その交差点で経路を入れ替えれば、経路の長さを変えずに、時計回りの巡回路が作れる。したがって、最短の時計回りの巡回路を構成すればよい。周上の 2 点間の最短経路は、その 2 点を結ぶ線分、1 点と多角形の頂点を結ぶ線分、および多角形の頂点同士を結ぶ線分で構成される。これらのうち多角形の外に出ない線分を残して、最短経路を求める。

問題 K: New Year Festival

イベントのスケジュールを決める問題である。イベントごとにそのイベントを開始する時刻の関数としてコストが与えられている。コスト関数は連続な区分線形関数になっており、入力ではその頂点が与えられる。また、それぞれのイベントには所要時間が設定されている。二つのイベントの開催時間が重ならないようなスケジュールのうち、コストが最小となるものを求める問題である。

あるイベントが終わると、隙間を空けず、直ちに次のイベントを開始することができる。このように、隙間なく連続して行われているようなイベントの並び全体を「イベント列」と呼ぶことにしよう。一般に、スケジュールはいくつかのイベント列に分解できる。次の理由から、最適解（の一つ）では、どのイベント列についても、イベント列を構成するイベントの少なくとも一つの開始時刻が、そのイベントのコスト関数のいずれかの頂点上にある。そうでないイベント列を含む最適解があるとする。どのイベントの開始時刻も頂点上にない場合、全てのイベントの開始時刻はコスト関数が線形な区間にある。したがって、コスト関数の傾きを考えることができる。コスト関数の傾きを合計して 0 でなければ、イベント列全体をずらすことで、合計コストをより小さくでき、最適性に反する。傾きの合計が 0 であれば、合計コストを変えずに、いずれかのイベントのコスト関数の傾きが変わるまで、つまり、いずれかのイベントの開始時刻が頂点上に来るまでイベント列全体をずらすことができる。ずらしている途中で隣のイベント列とぶつかったときは、連結した列全体について同様に考えることができる。

コスト関数の頂点のうち、そのイベントの開始時刻になっている頂点を「イベント付き頂点」と呼ぶことにする。時刻の早い順にイベント付き頂点を選びながら、イベント付き頂点の間にイベントをうまく配置すれば、合計コストが最小となるスケジュールが得られる。想定解法では、直前のイベント付き頂点と既に配置したイベントの集合との組を状態として、最小の合計コストを求める動的計画法を用いる。状態の遷移では、次のイベント付き頂点と、間に追加するイベント集合の選び方の全ての組合わせを考えてコストが最小のものを選ぶ。これをそのまま計算すると時間がかかるので、前計算で全ての頂点の組と、全てのイベントの選び方について、その頂点間に選んだイベントを全て配置する最小の合計コストの表（頂点間にイベントを配置するコストの表）を求めておく。この表を使えば、この動的計画法の部分は $O(M^2 3^n)$ になる。ここで、M は全てのコスト関数の頂点数の合計である。

頂点間にイベントを配置するコストの表も次のように $O(M^2 3^n)$ 時間で計算できる。最初の考察から、二つの頂点から内側に向かって隙間なくイベントを配置したスケジュールの中に最適解がある。そこで、さらに、頂点とイベントの集合が与えられたとき、頂点から前（後）に向かって与えられたイベントを隙間なく配置するときの最小コストを前計算しておく。これは $O(Mn2^n)$ で計算できる。各 2 頂点間で、この結果を参照しながら、各イベントを前に付ける、後ろに付ける、選ばないの 3 通り全てを試す。これにより、選んだイベントを 2 頂点間に配置する最小コストが計算できる。