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	0	0	6	3	5	11	8	7	0	0
チーム数累計	0	0	0	0	6	9	14	25	33	40	40	40

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
正答数	40	40	11	20	0	0	16	1	4	33	2
失敗数	0	0	3	11	1	4	9	5	4	3	3
提出数	47	101	24	101	1	11	85	43	16	99	13

失敗数：解答を提出したが正解にまで到らなかったチームの数
提出数：正誤の如何にかかわらず、提出されたプログラムの総数

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	J	C	D	F	I	G	H	K	E
正答数	40	40	33	11	20	0	4	16	1	2	0

今年のコンテストもオンラインでの開催となってしまった。開催 1ヶ月前までは開催形態が決定されていなかったが、審判団では当初からオンライン開催を想定して問題を準備した。オンラインにするかオンサイトにするかで問題の難易度や出題傾向を大きく変える必要がある。オンライン開催では、Wikipedia など多くのインターネット上の資料を参照でき、ソースコードに貼り付けられる限り既存のライブラリを利用できる。したがって、資料を参照しても難しい部分が残るような問題でなければならない。また、複数の計算機を使ったり資料を参照することで解答時間が短縮される可能性を考慮して、難しい問題を出題する必要がある。さらに、難しい問題が 1 問だけでは得意ジャンルが出題されるかどうかが上位チームの順位に大きく影響する可能性がある。これらを考慮すると、数問は難しい問題を準備する必要があった。

コンテストでは、どのチームも正解できない問題が 2 問残る結果になった。問題 H, K, E の正答数が少ないのは想定通りとはいえ、正答数が最高で 7 問というのは問題セットが難し過ぎたと言わざるを得ない。7 問正解の 6 チームは全て、問題 A, B, C, D, G, J の 6 問に正解したうえで、最後に問題 I か K に正解するという結果になっており、もう少し中盤の問題から難しい問題への難易度のバリエーションが多い方がよかっただろう。

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: Loop of Chocolate

重なりを考えずに n 個の球の体積を求め、そこから重なった部分の体積を引けばよい。重なった部分は、球を平面で切った断片 (spherical cap) を二つくっつけた形になる。

全チームが解ける一番簡単な問題を意図した。問題 A には珍しい幾何ではあるが、球や spherical cap の体積を求める式は問題文で与え、さらに、例外的な場合が起こらないように条件を設定した。この問題は、全チームが正解した。

問題 B: Lottery Fun Time

どの当選番号も下 2 桁が異なるという条件が鍵となる。100 種類の下 2 桁の番号ごとに、手持ちの抽選券の枚数と、そのうち下 4 桁が共通する抽選券の枚数の最大値をあらかじめ計算しておく。そのうえで、例外的な場合に注意しながら、起こり得る場合を全て試せばよい。

例外的な場合が多く、問題 B にしては難しい問題だった。実際、最終的に上位になったチームも含め、多くのチームが誤答を提出した。

問題 C: Reversible Compression

圧縮文字列を生成する途中状態を頂点とする DAG の最短路を探す問題として定式化する。途中状態では圧縮文字列のプレフィックスである圧縮文字列片が作られる。その圧縮文字列片を展開すると圧縮前文字列の先頭 i 文字が復元でき、反転した圧縮文字列片を展開すると末尾 j 文字が復元できるとき、この圧縮文字列片を持つ途中状態を頂点 (i, j) で表す。圧縮文字列の生成過程では、圧縮文字列片は 2 文字ずつ長くなる。その 2 文字を辺とする。このような DAG で、$(0, 0)$ から (n, n) に至る最短路を探す。辺のコストが全て同じなので、そのような経路は幅優先探索で効率よく求めることができる。

$(n+1)^2$ 個の頂点の範囲を重複なく探索できていれば、計算量が最善でないプログラムでも正解できるタイムリミットを設定した。

問題 D: Wireless Communication Network

木を構成する中で最も標高の高い通信局をその木の根とすることにしよう。そうすると、直径が最大となるように構成した木では、直径に対応する道 π は必ず根を通る。もし、π が根 r を通っていなければ、π 上で最も標高が高い通信局を u、π 上で u の直前か直後の通信局 v としたとき、π の (u, v) を除く全ての辺と、辺 (u, r), (r, v) を持つように木を構成できる。これには、木を構成する途中の辺 (u, v) を作るステップで、代わりに辺 (u, r) と辺 (r, v) を作ればよい。当然、このように作られた木には π より長い道が存在する。

通信局に左から順に番号を付けたとき，番号が $[l, r]$ の区間の山頂をつなぐ木の直径の最大値 $\mathit{diameter}(l, r)$ は次の式で与えられる。

$$\mathit{diameter}(l, r) = \max \begin{cases} \mathit{diameter}(l, m-1) + 1 \\ \mathit{diameter}(m+1, r) + 1 \\ \mathit{height}(l, m-1) + \mathit{height}(m+1, r) + 2 \end{cases}$$

ここで，m は $[l, r]$ の区間で最も標高が高い通信局の番号、$\mathit{height}(l, r)$ は $[l, r]$ の区間の通信局をつなぐ木の高さの最大値である。ただし $l > r$ のときは、$\mathit{diameter}(l, r) = \mathit{height}(l, r) = -\infty$ とする。上

式の max の第 1, 2 式が、上記の方法で、それぞれ m の左側と右側の木の直径の道の途中に m を挿入した道に対応する。

この計算では、様々な区間に対して何度も m を求める必要があるが、動的計画法を工夫して m の計算を減らすことができる。あるいは、セグメント木を使って m を高速に求めることもできる。

問題 E: Planning Railroad Discontinuation

与えられたグラフの最小全域木を求める問題だが、グラフが大きすぎて Kruskal のアルゴリズムを直接使うことはできない。うまくグラフの対称性を利用して、同じ計算をまとめることが鍵となる。

まず一つの街の中だけの最小全域木を作ることを考える。地下鉄の路線はそれほど多くない ($m \leq 10^5$) ので、これは Kruskal のアルゴリズムで間に合う。この一つの街の中だけの Kruskal のアルゴリズムと、新幹線路線も含めたグラフ全体の Kruskal のアルゴリズムで変化がある部分を考えよう。新幹線駅を含まない木とそれ以外の木は地下鉄の路線で連結するしかないので、一つの街の中でも国全体でも変わらない。したがって、あらかじめこのような辺を縮約しておくことで、すべての駅が新幹線駅であるような場合に帰着される。また、新幹線路線も含めたグラフ全体では、地下鉄の路線を使うよりも新幹線を使って隣街を経由する方がコストが小さくなることがある。これは、この街の地下鉄のベースコストが隣街のベースコストより大きく、かつ、連結しようとしている地下鉄駅 u,v 間のコスト $d(u,v)+b$ と隣街へのコスト a の間に

$$d(u,v)+b > a$$

が成り立つときに起こる。Kruskal のアルゴリズムでは、コストが小さい辺から順に森に追加するので、途中まで地下鉄路線だけを対象に Kruskal のアルゴリズムを適用し、$d(u,v)+b > a$ であるような辺を追加しようとした時にアルゴリズムを止める。この時連結されずに残った木の間は新幹線を使って隣街を経由した方がよい。この時追加する新幹線路線の数は、残った木の数に等しい。そして、新幹線で隣街とつなぐと、ベースコストが大きい方の街は忘れてもよい。

全ての街は同じ形の地下鉄網を持っているので、Kruskal のアルゴリズムのうち街の中で閉じた部分はどの街でも共通の処理になる。街ごとに違うのは、どこでアルゴリズムを止めるかである。そこで、前処理として、一つの街の中だけの最小全域木を求めておく。その途中の各ステップで、各コスト以下の辺まで森に加えた時の合計コストや、連結されずに残った木の数を記録した表を作っておく。街の中のコストや街をつなぐ新幹線路線の数はこの表を参照して求めることにして、街間を新幹線路線の 1 路線あたりのコストが低い順に連結すれば、制限時間内に正解を出力することができる。

この問題は、難しい問題の枠で出題した。残念ながら 1 チームにも解かれることはなかった。

問題 F: It's Surely Complex

与えられた素数 p と正整数 n について、$0 \leq a,b \leq n$ である $a+bi$ のうち a も b も p の倍数でない数をすべて乗じた数を $\bmod p$ で求める問題である。

想定解法では、$a=b$ の場合と $a \neq b$ の場合に分けて積を求める。$a=b=k$ の場合、$a+bi$ は $k(1+i)$ と表せる。数列 $d_k = (k \bmod p)(1+i)$ には周期 p で同じ数が現れるので、1 周分の積を求めてそれを周期数分だけ掛ければよい。k が p の倍数の場合は掛けないことに注意すると、$m = \lfloor n/p \rfloor$ を端数を捨てた周期の数、$r = n \bmod p$ を端数として、$a=b$ である数の積は

$$((p-1)!)^m (1+i)^{m(p-1)} \times r!(1+i)^r$$

と $\bmod p$ で一致する。

$a \neq b$ の場合、$a + bi$ を掛けるときは $b + ai$ も掛けることになり、

$$(a + bi)(b + ai) = (a^2 + b^2)i$$

である。したがって，$0 \leq j \leq p - 1$ である各 j について、$a^2 + b^2 \equiv j \pmod{p}$ となる組 $\langle a, b \rangle$ の個数 c_j が分かれば、$a \neq b$ である $a + bi$ の積が計算できる。

この問題では、c_j の求め方にも工夫が必要になる。$\bmod p$ の周期性を使っても、単純に $0 \leq a, b < p$ の空間を全探索すると $O(p^2)$ の計算量になり間に合わない。そこで、畳み込み演算に持ち込み、高速フーリエ変換（FFT）を使って $O(p \log p)$ の計算量で計算する。具体的には、e_j を $a^2 \equiv j \pmod{p}$ である $a (0 \leq a < p)$ の個数として、前半が e_j、後半が 0 の $2p$ 要素のベクトル $e = (e_0, \ldots, e_{p-1}, 0, \ldots, 0)$ の畳み込み $e * e$ を考える。$e * e$ の結果の第 k 要素は

$$\sum_{k_1 + k_2 = k} e_{k_1} e_{k_2}$$

となり、第 j 要素と第 $p + j$ 要素から c_j を求められる。ただし、これらの数には $a = b$ である場合も含まれていることに注意が必要である。

問題のタイトル通りの複雑な問題で、残念ながら正解チームはなかった。

問題 G: Genealogy of Puppets

この問題の想定解法は少し変わった動的計画法になる。漸化式の引数に何を選ぶかが鍵となる。

まず、最新型の人形 1 個だけからなる木を作り、残った人形を発売日が遅い順に追加して、最終的に全ての人形をつなげる。この途中に、追加する人形を既存の木につなげず、一時的に木を複数にすることを許す。つまり、途中状態では森となる。

人形を追加するときには、まず、その人形に最終的にいくつ子を吊るすことにするかを決める。子を吊るすことにした足を「スロット」と呼ぼう。子を片足にしか吊るさない時は左右の足を区別しない。最初はスロットは全て空いている。次に、追加する人形を既存の木の空きスロットのどこかに連結し葉にするか、新しい木の根にする。さらに、追加した人形にスロットがあり、木が複数あるとき（追加する人形を新しい根にした場合も含む）は、別の木の根を追加した人形の空きスロットに接続してもよい。ただし、子を持つ人形では少なくとも一つ発売日が遅い子を持つという条件から、スロットの数を 1 以上としたときは、少なくとも一つは既存の木を追加する人形のスロットに接続しなければならない。

この手順で人形を k 個追加したときに作られる異なる森の数を、木の数と、空きスロット数ごとに分けて計算する。この計算では、$k - 1$ 個追加したときの数をもとに、

- 追加する人形のスロットの数
- 追加する人形を既存の木に吊るす場合と新しい木の根にする場合
- 追加する人形のスロットに接続する既存の木の数

の全ての可能な組合わせで異なる森の数を計算して合計する。

このようにして計算して、最後の人形を追加した後に木の数が 1 で、空きスロットの数が 0 である森の数が答えになる。

問題 H: Cancer DNA

この問題の想定解法は乱択アルゴリズムである。長さ n の DNA のうち、どれかのパタンにマッチするものの個数を G とする。DNA に現れる文字は `A`, `G`, `C`, `T` の 4 種類なので、求める確率は $G/4^n$ である。これを直接乱択で求めると、G が非常に小さいときに相対誤差が大きくなり、少ないサンプルで相対誤差 5%以内の要件を満たすことができない。

そこで、重複を許してどれかのパタンにマッチする、つまり、同じ DNA が二つのパタンにマッチする場合には 2 個と数えたときのマッチする個数を U とし、U を使って次のように式を変形する。

$$\frac{G}{4^n} = \frac{U}{4^n}\frac{G}{U}$$

U はパタンに現れる?の数から簡単に計算できるので、G/U が求まればよいことになる。これは、いずれかのパタンにマッチする DNA をランダムに十分な数生成して、それが二つ以上のパタンにマッチするかどうか判定することで近似できる。

上述の「十分な数」は Hoeffding's inequality などの確率集中不等式を使って理論的に評価することができる。具体的には、$O(\epsilon^2)$ 個の DNA をランダムに生成すると、加法的な ϵ の誤差を許して G/U を近似できる。$G/U \geq 1/m$ であるから、$\epsilon = 0.05/m$ と定めることにより、G/U を相対誤差 5%以内で高い確率で近似することができる。このように、相対誤差を許して高い確率で計算するアルゴリズムのことを FPRAS といい、上述のアルゴリズムは DNF 式の充足割当てに対する FPRAS の特殊ケースとなっている。

問題 I: "Even" Division

与えられたグラフを、それ以上分割できなくなるまで偶数個の頂点を持つちょうど二つの連結部分グラフに分割し続ければ解が得られる。しかし、頂点数の制限がなくても、グラフを二つの連結部分グラフに分割する方法は自明ではない。条件を満たす分割が可能なときに必ず分割できる方法を見つけることがこの問題の核心である。

まず、グラフが木の場合に限定して考えよう。この場合は、どれでもいいので辺を 1 本選ぶと、その辺を取り除くことで二つの連結部分グラフに分割できる。さらに、木に偶数次数の頂点 v があれば、v から出る辺のどれかを切れば偶数個の頂点を持つ部分木が作れる。もし、どの切り方でも部分木の頂点数が奇数になるとすると、v をとり除いてできる偶数個の部分木の頂点数の合計が偶数になり、v を含めた元の木全体の頂点の数が奇数になってしまう。一方、全ての頂点の次数が奇数のときは、どの辺をとり除いても頂点数が奇数の部分木に分かれる。そうでなければ、取り除く辺 e の一方の先にある全ての頂点の次数を合計すると偶数になってしまう。しかし、その部分木には辺 e の一端しか接続していないので、奇数でなければならない。したがって、全ての頂点の次数が奇数のときはこれ以上分割できない。

グラフが木でない場合は、全域木をとることで木にして考える。このとき、全域木の作り方によっては全ての頂点が奇数になることがある。そうなったときは、全域木に使われている辺を別の辺と入れかえることで、次数が偶数の全域木にすることができる。

計算量についても注意を払わなければならない。分割のたびに全域木を作ると $\Omega(nm)$ 時間かかり、制限時間を超えてしまう。しかし、一度作った全域木を分割後も壊さないようにして使いまわせば、計算量を下げられる。

問題 J : The Cross Covers Everything

まず点を x 座標で整列し、交差点の左下になり得る頂点を x 座標が小さい点から順に探す。そのような各点に対して、今度は交差点の右上になり得る頂点を x 座標が大きい点から順に探す。これを素直にプログラムにすると計算量が $O(n^2)$ になり間に合わない。そこで、あらかじめ、x 座標が大きい点から順に調べたとき、最大値（最小値）を更新する点だけからなる増加部分列と減少部分列を求めておく。これを二分探索することで、交差点の右上になり得る点の数を一度に計算できる。

この問題では、競技中に問題訂正を行った。選手にはご迷惑をおかけしたことをお詫びする。

問題 K: Distributing the Treasure

この問題では、チームメンバーが不満を顕にする手前の、心の中で「うらやんでいる」関係を利用する。A が B をうらやんでいるとは、A の価値基準で A に配られた宝物の価値の合計よりも B に配られた宝物の価値の合計が大きいが、B に配られた宝物のどれを取り除いても A に配られた宝物の価値の合計以下になることを言う。A が B をうらやんでいるとき、さらに宝物を 1 個 B に分配するとその価値にかかわらず A は不満を顕にする。

宝物の価値の順序はメンバーによらず一通りに決まっている。想定解法の方針は、宝物を価値の高いものから降順に分配していき、誰にも分配できなくなったら、各メンバーに既に配られた宝物をそっくり別のメンバーと交換する（以下ではプレゼント交換と言おう）というものである。当然だが、宝物を分配するときには、誰かにうらやまれている人には分配することができない。つまり、全員がだれかにうらやまれると、プレゼント交換が発生する。プレゼント交換が発生する状況では、うらやんでいる関係がサイクルになっているはずである。このサイクルに沿ってプレゼント交換を繰り返すと、だれからもうらやまれていないメンバーができる。プレゼント交換をするたびに、交換に参加した各メンバーについて、そのメンバーにうらやまれるメンバーが最低一人減るからである。

なお、チームメンバーの人数 n が宝物の数 m 以上のときは、だれにも 2 個以上配らないようにするだけで誰も不満を言わないようにできる。したがって $n < m$ の場合を考えればよい。この問題では $nm \leq 2 \times 10^5$ という制約があるので、$n < m$ の場合 $n < 447$ という比較的小さい n についてだけ、プレゼント交換が発生する状況を想定すればよい。

上記の解法では、サイクルの検出は $O(n^2)$ 時間で可能であり、サイクルを解消する回数は全体を通して nm を超えないことが示せるため、全体では $O(n^3m)$ 時間で解ける。実際にはこの見積りはかなり甘く、制限時間に十分間に合う。理論的には計算量がより小さい、$O(n^2(m-n))$ の解法も存在する。その基本的なアイデアは、メンバーとメンバーに配られた宝物の集合を別の頂点とするグラフを作り、最大重み完全マッチングを保持するというものである。このグラフでは、宝物の集合（プレゼント）から所有者への辺と、プレゼントをうらやんでいる人からプレゼントへの辺を張る。宝物をだれかのプレゼントに追加してうらやむ人が増えると、サイクルが生まれることがあるが、交換後の重みが最大となるようなサイクルに沿ってプレゼント交換すれば最大重み完全マッチングを正しく更新でき、これは DAG 上の最長経路問題を解くことで $O(n^2)$ 時間で計算できる。