電気電子情報工学実験II (b)
実践的・競技プログラミング 第2回

競技プログラミングの問題を解くための考え方

廣田悠輔 *)
*) y-hirota@u-fukui.ac.jp

はじめに

- 本資料は，競技プログラミング（以下，競プロ）の問題を解くための考え方について説明する.
- 本資料の説明はあくまで考え方の一例であるので，必ずしも説明の通りに考える必要はない.
- 自分なりの考え方・解き方の確立を目指すこと.

基本的なプロセス

自然言語（日本語や英語）で書かれた問題文

↓ 回答に関係のある部分の抽出

問題の本質部分

↓ アルゴリズムの組み合わせへの落とし込み【難】

アルゴリズムの組み合わせによる回答の表現

↓ コーディング

回答ソースコード

問題例（「それにつけても金の欲しさよ」）

問題文

お金が入った袋が n 個ある．第 $1, 2, \ldots, n$ 番目の袋には，それぞれ $A_1, A_2, \ldots, A_n$ 円のお金が入っている．あなたは n 個の袋のうち，m 個を自由に選んで受け取ることができる．あたたが受け取る金額の合計が最大になるように袋を選んだ時の，受け取り金額を出力するプログラムを作成せよ．

制約

- $1 \leq m \leq n \leq 10^5$
- $1 \leq A_i \leq 10^3$

出力

受け取る金額の合計が最大になるように袋を選んだ時の，受け取り金額を出力せよ．

問題の本質部分の抽出

問題文

お金が入った袋が n 個ある．第 $1, 2, \ldots, n$ 番目の袋には，それぞれ $A_1, A_2, \ldots, A_n$ 円のお金が入っている．あなたは n 個の袋のうち，m 個を自由に選んで受け取ることができる．あたたが受け取る金額の合計が最大になるように袋を選んだ時の，受け取り金額を出力するプログラムを作成せよ．

↓ 回答に関係のある部分の抽出

既知の値

$n, m, A_1, A_2, \ldots, A_n$　（いずれも正整数）

求めるべき値

$A_1, A_2, \ldots, A_n$ から m 個を選んで総和をとったときの最大値

アルゴリズムの組み合わせへの落とし込み [1/9]

既知の値

$n, m, A_1, A_2, \ldots, A_n$　（いずれも正整数）

求めるべき値

$A_1, A_2, \ldots, A_n$ から m 個を選んで総和をとったときの最大値

【やるべきこと】

- 様々なアルゴリズムを検討して，問題を解く方法の候補を考える.
- 問題を解く方法の候補の計算量や記憶領域を大まかに見積もる.
  - 計算資源制約を満たさない候補は除外する.
- 残った候補の中から，1つを選んでコーディングの対象とする.

アルゴリズムの組み合わせへの落とし込み [2/9]

既知の値

$n, m, A_1, A_2, \ldots, A_n$　（いずれも正整数）

求めるべき値

$A_1, A_2, \ldots, A_n$ から m 個を選んで総和をとったときの最大値

愚直な方法

- $A_1, A_2, \ldots, A_n$ から m 個を選ぶ組み合わせすべてについて，その総和を求める.
- 組合せごとの総和の中から最大値を選んで出力する.

✓ 計算量の概算

- $n, m, A_1, A_2, \ldots, A_n$ の読み込み.　→ $O(n)$
- 全組み合わせについてそれぞれ総和計算.　→ $O\left(\binom{n}{m} m\right)$
  - $1 \leq m \leq n \leq 10^5$ なので，時間制約を満たせないと予想できる.

アルゴリズムの組み合わせへの落とし込み [3/9]

【1】より扱いやすい等価な別の問題への読み替え

求めるべき値

$A_1, A_2, \ldots, A_n$ から m 個を選んで総和をとったときの最大値

問題の読み替え

求めるべき値

$A_1, A_2, \ldots, A_n$ の大きい方から m 個の値を選んだときの総和

アルゴリズムの組み合わせへの落とし込み [4/9]

【2】問題の分割

- **簡単化された問題を考える：**
  - $A_1 \geq A_2 \geq \ldots \geq A_n$ という都合の良い状況が成立すると仮定する.
  - この仮定の下では，$A_1, A_2, \ldots, A_m$ を選んで総和を取れば，それが求めるべき値となる.

✓ 計算量の概算

- $n, m, A_1, A_2, \ldots, A_n$ の読み込み. → $O(n)$
- $A_1, A_2, \ldots, A_m$ の総和計算. → $O(m)$

アルゴリズムの組み合わせへの落とし込み [5/9]

【2】問題の分割（続き）

- **簡単化された問題への帰着：**
  - 実際の問題は，$A_1 \geq A_2 \geq \ldots \geq A_n$ という都合の良い入力が与えられるわけではない.
  - では，どうすれば元の問題を，簡単な問題に変換できるか？
  - この問題は，入力 $A_1, A_2, \ldots, A_m$ を降順ソートすれば，簡単化された問題に帰着できる.

アルゴリズムの組み合わせへの落とし込み [6/9]

【2】問題の分割（続き）

- **元の問題を，以下の2つの小問題の組み合わせと考える：**
  - 与えられた n 個の入力を降順にソートする問題.
  - 降順ソートされた値から大きい順に m 個を選んだときの総和を求める問題.

✓ 計算量の概算

- $n, m, A_1, A_2, \ldots, A_n$ の読み込み. → $O(n)$
- $A_1, A_2, \ldots, A_n$ の降順ソート. → $O(n \log n)$ から $O(n^2)$
- $A_1, A_2, \ldots, A_m$ の総和計算. → $O(m)$

✓ 計算量 $O(n^2)$ では時間制限を超過するが，計算量 $O(n \log n)$ のソートアルゴリズム（マージソート等）を使えば制限時間内で可能と見積もられる.

アルゴリズムの組み合わせへの落とし込み [7/9]

- ✓ 以上の思考プロセスは，回答者が組合せやソートについて理解している場合のもの.

- ➢ 競プロでは（あるいは競プロに限らず），その場で使える基本アルゴリズムやデータ構造，数学的知識が少ないと著しく不利.
  - コンテスト時間内に閃くことは稀.
  - 事前の勉強が重要.

- ➢ 問題を適切に読み替えたり分割するには，上記の知識に加え，ある程度の経験や慣れが必要.
  - 競プロ関連書籍に掲載されている定番テクニックを知る.
  - 競プロの過去問を解いて経験を積む.
  - 問題の入力例を紙上で処理してその流れを追い洞察する.
  - 日頃から複雑事象を細かなプロセスに分解して考える癖をつける.

アルゴリズムの組み合わせへの落とし込み [8/9]

- ✓ 計算資源制約を守りつつ正答できそうな方法が複数存在する場合：
  - 簡単に実装できる方法を選択する.
  - 無理に最も効率的なアルゴリズムを使う必要はない.

アルゴリズムの組み合わせへの落とし込み [9/9]

- ✓ 競技プログラミングでは必ず適切な解法が存在すると仮定できる.

- 入力サイズから使用すべきアルゴリズムが推定できる場合がある.
  - 【例1】制限時間が 1 [sec.] で入力 n の最大値が 10^5 のケース：
    - 計算量 $O(n^2)$ 以上では時間制限を超過.
    - 計算量 $O(n \log n)$ 以下の解答例が存在すると推測可能.
  - 【例2】入力 n の最大値が 10 といった小さい値のケース：
    - 2^n 回の処理が許容される.
    - 探索問題であれば全探索も検討の範囲内となる.

  - ✓ 邪道だが非常に強力なヒントになる.

コーディング

- 素早く正確にコードを記述する.
- アルゴリズム完成後にコーディングを開始する必要はない.
  - アルゴリズムを検討しながら，並行してソースコードの大枠を記述しても良い.
  - 簡単な問題では，問題を読んだ時点で使うべきアルゴリズムは半ば見えているので，並行して記述する方がより早いこと多い.
- 使用する型の表現可能範囲に注意.
  - 例えば，符号付き32ビットの表現範囲を超えることはないか.
  - 浮動小数点数を使う場合，精度は十分か.
- コンパイラの警告に注意.
  - 変数の未初期化などの警告が出ている場合は無視しない.