第 2 章

全探索と計算量の見積もり

2017-10-04 Wed

概要

可能性を全て試すプログラムと，その計算量の見積もりについて学ぶ．今回から一部の問題には考察課題が指定されている場合がある．ソースコードを提出する際に，指定された考察も添えること．

2.1 問題の大きさの把握 (Constraints)

例題 **Min, Max and Sum (再掲)** (AOJ)

n 個の整数 $a_i (i = 1, 2, ...n)$ が与えられるので，それらの最小値，最大値，合計値を表示せよ．
http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=ITP1_4_D

入力の制約: $0 < n \leq 10\,000,\ -1\,000\,000 \leq a_i \leq 1\,000\,000$

実行時の制約: Time Limit : 1 sec, Memory Limit : 65536 KB

さて，求められているプログラムは「制約を満たす範囲でどのような入力にも正しい出力を行う」ものである．従って，扱う入力の範囲を理解し，適切な解法を設計する必要がある．問題の範囲と有効な解法の関係は，先学期のプログラミング演習では意図的に扱わなかったが，今学期の本演習では主題に近い重要な点である．

■数値の表現 通常，C++ で整数は `int` 型の変数で表現する．一方，この演習の iMac 環境では，表 2.1 のように，`int` 型や他の整数型で表現できる値の範囲には限りがある．

配列の各要素は，$\pm 1\,000\,000$ ($\approx 2^{20}$) なので，`integer` で表現可能な範囲に余裕を持って収まっている．一方，「合計」はどうだろうか．個数 n の上限ですべて要素が最大値をとったとすると

$$1\,000\,000 \cdot 10\,000 = 10^{10} > 2^{31} \approx 2 \cdot 10^9$$

と 32bit 整数で表現できる範囲を超える．つまり，合計を表す変数には `long long` 型を使う必要がある．このことをプログラムを書き始める前に把握できるようになることが，目標の一つである．[*1]

[*1] このような計算をスラスラと行うために $2^{10} = 1\,024 \approx 10^3$ を暗記しておくことを勧める．

表 2.1　この資料が想定する環境での，符号付き整数の表現

type	bits	下限	上限	備考
int	32	-2^{31}	$2^{31}-1$	約 20 億
long	32/64			使用を勧めない
long long	64	-2^{63}	$2^{63}-1$	C++11 から標準型，以前は gcc 拡張
__int128_t	128			gcc 拡張，portable でないが強力

各型で表現可能な範囲は環境によって異なりうる．使用中の環境については，以下のようなプログラムを用いて確認できる．numeric_limits の文法は割愛するが，興味のあるものはテンプレートクラスと標準ライブラリについて調べると良い [4].

C++

```cpp
#include <limits>
#include <iostream>
using namespace std;
int main() {
  cout << sizeof(int) // バイト数
       <<' '<< numeric_limits<int>::digits // 符号以外の bit 数
       <<' '<< numeric_limits<int>::digits10 // 十進桁数
       <<' '<< numeric_limits<int>::min()
       <<' '<< numeric_limits<int>::max()
       << endl;
  cout << sizeof(long long)
       <<' '<< numeric_limits<long long>::digits
       <<' '<< numeric_limits<long long>::digits10
       <<' '<< numeric_limits<long long>::min()
       <<' '<< numeric_limits<long long>::max()
       << endl;
}
```

```
$ ./a.out
4 31 9 -2147483648 2147483647
8 63 18 -9223372036854775808 9223372036854775807
```

なお表 2.1 には符号付き整数のみ記載したが，符号なし整数 (unsigned int など) についても，各自把握のこと．C++11 では#include<cstdint>すると，int32_t, int64_t などの型を使用可能である．将来はこちらを使用するべきだが，本資料では，int と long long を用いる．

■時間と空間の制約　各問題には，実行時の制約も付加されている．先の例題ではプログラムは，実行時間 1 秒未満で回答を終えること，利用メモリは 64MB 未満であることが求められている．

プログラムの実行時間やメモリの使用量は，(1) 実行環境，(2) 問題の大きさ，(3) アルゴリズムによって大きく変わりうる．

それぞれ簡単に議論すると，AOJ の実行環境 (1) はサイト内の文書に記載されているが，演習室の端末よりも高速なので通常は気にする必要はない．つまり手元で十分効率的に動くことを確認すれば良い．

問題の大きさ (2) は，たとえばこの例題では数値列の長さが相当する．数値をすべて標準入力から読み配列に格納するとすると，長さに比例する時間がかかると予想される．たとえば，数値列が長さ 1 であればすぐに終わるが，10 000 であれば (正確に 1 万倍かどうかはさておき) もっと時間がかかるだろう．使用メモリは (3)，数値を整数型 (4 byte) で表すと $4 \cdot 10\,000$ byte であり，入力をすべて配列に格納しても 1MB に達しない，微々たる量であることが分かる．(プログラムが使用するメモリ量は他にもあるが，この演習では問題の大きさで変化しうる部分のみを考える．)

アルゴリズム (3) がどのような役割を果たすかが，もっとも重要な話題であり，今後時間をかけて取り扱う．

2.2 組み合わせを試す (generate and test)

先週の問題で要素を全て調べて最大値を求めたように，コンピュータが得意な解法は全ての可能性を力づくで試すことである．実際に多くの問題をそれで解くことが出来る．全ての可能性を列挙するには，for 文を用いたり，再帰を用いたりする．

問題 **Tax Rate Changed** (国内予選 2014)

2 つの商品の消費税率変更前の税込合計価格を元に， 新消費税率での税込合計価格が最大いくらになるかを計算するプログラムを作って欲しい． 1 円未満切り捨て等の細かい条件は問題文を参照のこと． http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=1192&lang=jp

制約 (抜粋): $10 < s < 1000$， 商品の税抜価格は 1 円から $s - 1$ 円のすべてを考慮に入れる

Time Limit : 8 sec, Memory Limit : 65536 KB

(1) 2 つの商品の価格の候補の組み合わせを全通り試す．(2) 候補が，変更前の税率で合計価格になっているかを調べる．そうでなければ捨てる (3) 変更後の税率での合計価格を計算する．最大値なら記憶する．という方針で作ってみよう．

仮に税込みの価格を計算する `tax`(*rate, price-without-tax*) という関数があったとすると，次のような構造になるだろう．問題文では小文字の s を使っているが，本資料のプログラムでは (i, j, k などの変数と区別して問題文由来である目印に) 大文字で表記する．

C++

```
1  int maximum = 0;
2  for (int i=1; i<S; ++i)
3    for (int j=i; j<S; ++j)
4      if (tax(X,i)+tax(X,j) == S) // (*)
5        maximum = max(maximum, tax(Y,i)+tax(Y,j));
```

なおこの例題はさらに，繰り返し回数を減らすための工夫を検討することもできるが，今回は必須ではない．

■計算量の簡単な検討 (estimation of the number of trials)　上記のプログラムで 4 行目 (*) が最大何回実行されるか，考察しよう (厳密に求めなくて良い)．回数を (S に依存するので)，$f(S)$ と表すとする．2 行目の `for` ループが S 回，3 行目の `for` ループが最大 S 回繰り返すので，$f(S) \leq S^2 \leq 1\,000^2 = 10^6$ である．

表 2.2　C++ で，1 秒間で実行可能な演算回数の目安 ([3] を改変して転載)

1 000 000	余裕を持って可能
10 000 000	たぶん可能
100 000 000	とてもシンプルな演算なら可能

この回数 (10^6) を表 2.2 に当てはめると，(tax 関数が効率的に実装されていることを前提に) 余裕を持って，1 秒間で実行可能と分かる．[*2]表の数値は，実行環境に合わせて経験的に測定する必要がある．目安として，最近の多くのコンピュータの CPU は 1GHz ($= 10^9$) 程度で動作しているので，CPU が 100 サイクル以内に実行できる演算なら，1 秒間に 10^6 回実行可能と期待できる．

本演習の範囲では，この表を信じて問題ない場合がほとんどであるが，現実の計算機は複雑な装置であるから，正確な実行時間の予想は簡単ではない．たとえばメモリ参照や new/delete は，算術演算よりそれぞれ 10 倍，100 倍以上に遅い場合がある．様々な条件が影響するため，予測のためには何らかのキャリブレーションが必要である．また，C, C++ 以外の言語では，実行により時間がかかることも多い．

■複数データセットの入力　この問題ではデータセットが複数与えられうる．すなわち，データが与えられる間はそれを読み込んで解を出力し，データが終了したらプログラムを終了させる必要がある．「入力の終わりは，空白で区切られた 3 つのゼロからなる行によって示される． 」という条件と通常のデータセットでは X が正であることを活用し，以下の構造でプログラムを書くことを勧める．

C++

```
#include <iostream>
using namespace std;
int X, Y, S;
int solve() {
  ... // X,Y,S について，最大値を計算
}
int main() {
    while (cin >> X >> Y >> S && X>0) {
        cout << solve() << endl;
    }
}
```

while 文の条件の cin >> X >> Y >> S の部分は cin への参照を返し，bool にキャストされた際に，cin が正常状態かどうか，すなわち X, Y, S を正常に読み込めたかどうかを返す．入力ファイルが間違っていた (よくある場合は，タイプミスまたは違う問題の入力を与えた) 場合はここが false になって終了する．読み込めた場合に，X, Y, S は処理すべきデータセットの X, Y, S を読んだ場合と，入力終了の目印の空白で区切られた 3 つのゼロを読んだ場合の両方がある．前者の場合は正の整数であるはずなので，0 かどうかをテストして判別する．

■AOJ への提出　上記の方針でプログラムを作成し，AOJ で accept されることを確認せよ．

もし accept が得られなかった場合は手元での検証が可能である．まず，入力 (http://icpc.iisf.

[*2] 注: この見積もりは 1 つのデータセットについてのものである．一方，問題全体では「入力は複数のデータセットからなる」ものを 8 秒間で回答する必要があるため，計算時間はデータセット数の上限を考慮して見積もる必要がある．データセットの数について記述がない場合は，厳密には問題の不備であるが，10 個程度と考えて良い．

or.jp/past-icpc/domestic2014/qualify14_ans/A1) と正答 (http://icpc.iisf.or.jp/past-icpc/domestic2014/qualify14_ans/A1.ans) をダウンロードしておく.

```
$ ./a.out < A1 > my-output.txt
$ diff -u A1.ans my-output.txt
```

差があれば行数が表示される.

2.3 試行回数の見積もりと計算手順の改良

続いて，計算方法の微調整によって試行回数が異なる例題を試してみよう.

問題 **Space Coconut Crab** (模擬国内予選 2007)

宇宙ヤシガニの出現場所を探す．エネルギー E が観測されたとして，出現場所の候補は，$x+y^2+z^3=E$ を満たす整数座標 (x,y,z) で，さらに x+y+z の値が最小の場所に限られるという．x+y+z の最小値を求めよ．(x,y,z は非負の整数，E は正の整数で 1,000,000 以下，宇宙ヤシガニは，宇宙最大とされる甲殻類であり，成長後の体長は 400 メートル以上，足を広げれば 1,000 メートル以上 ← 同じ数値でも重要な制約と余計な情報の区別に注意)

http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=2012

考察課題: 下記を参照して，各方針の計算コストに関するグラフと考察を提出する.

まず，変数 x,y,z の組み合わせを全部試して間に合うかを考える．(間に合うならそれが一番実装が簡単なプログラムである．この問題は最小化したい式が x+y+z が単純な形をしているので試行錯誤なしに解くことができるが，全ての $f(x,y,z)$ で効率的な解法があるとは限らない．)

C++

```
int count = 0; // 計測用 global 変数
int solve(int E) {
  // 愚直な求め方
  int ans = ... // 最大値
  for (int x=0; x<=E; ++x)
    for (int y=0; y*y<=E; ++y)
      for (int z=0; z*z*z<=E; ++z) {
        ++count;
        if (x+y*y+z*z*z == E)
            ... // x+y+z が最小値なら ans を更新
      }
  return ans;
}
int main() {
  int E;
  while (cin >> E && E>0) {
    cout << solve(E) << endl;
```

```
18      }
19  }
```

前問同様に 8 行目が何回実行されるかを見積もってみよう．変数 x,y,z の動く範囲を考えると，それぞれ x:[0,E], y:[0,$\sqrt{E}$], z:[0,$\sqrt[3]{E}$] である．そのような (x,y,z) の組み合わせは，E の最大値が 1,000,000= 10^6 であることから，最大 $10^6 \cdot 10^3 \cdot 10^2 = 10^{11}$ である．表 2.2 を参照すると，制限時間が 8 秒であることを加味しても，この方針では間に合いそうにない．

実際に，main を以下のように差し替えると，各 E の値に対応する試行回数を表示することが出来る．

C++

```
int main() {
  int E=1000000;
  solve(E);
  cout << "count = " << count << endl;
}
```

まとめて計測する場合はループを作ると良い

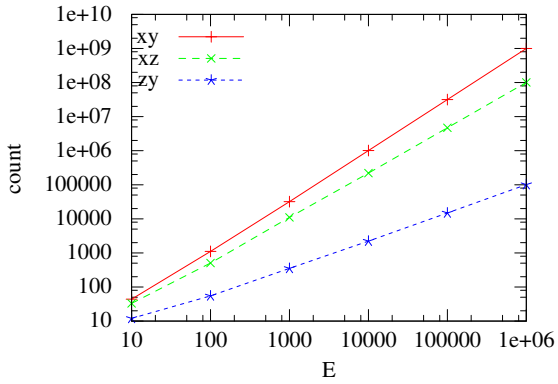
C++

```
int main() {
  int E[]={100,10000,1000000};
  for (int i=0; i<3; ++i) {
    count=0;
    solve(E[i]);
    cout << "E = " << E[i] << "count = " << count << endl;
  }
}
```

試行回数を減らすために，全ての (x,y,z) の組み合わせではなく，$x + y^2 + z^3 = E$ を満たす範囲のみを考えることにする．たとえば，x と y を決めると，$E - x - y^2 = z^3$ から z を決定できる (z が整数とならない場合，すなわち z=(int)round(pow(E-x-y*y,1.0/3)) として x+y*y+z*z*z==E とならない場合は無視して良い)．この方針で調べる種類は，(x,y) の組み合わせの種類である，$10^6 \cdot 10^3$ となる．この値はまだ大きいが，先ほどと比べると 1/100 に削減できている．上記と同様に，x と z を決めて y を求める，z と y を決めて x を求めることもできる．

一番良い方針が制限時間に間に合う範囲であることを確認し，AOJ に提出して確かめる．AOJ に提出する際は，count の表示は消しておくこと．

■考察課題　以下のグラフは，E を 10 から 10^6 まで適当に変えた時に，調べる組み合わせの種類数を実測したものである．たとえば xy は x,y,z の順に決めた場合を表す．



方針 xy と zy のプログラムを書き，いくつかの E に対する試行回数を出力し，グラフの数値と比較せよ．E の大きさと試行回数の関係に関する考察を，提出するソースコード内にコメントとして記入せよ．注意 このような数値実験の際は，ナマの数値を保存しておくこと．また転記の際は，手打ちではなく，コピーペーストを用いること (転記ミスを防ぐため)．プログラムの出力は次のように redirection を用いて，ファイルに保存する．

```
$ ./a.out > xy-e-vs-count.log
```

(オプション) グラフを自作して提出する．

(オプション) 試行回数とプログラムの実行時間の関係を調査せよ．たとえば横軸を試行回数，縦軸を実行時間として散布図を描く．プロセス全体の実行時間の測定は，`time` コマンドが利用可能である．

```
$ time ./a.out
```

関数毎の所要時間を測る方法には，一例として C++11 から標準となった `chrono` ライブラリを紹介する．C++11 の機能を使うには，`g++ -Wall -std=c++11 filename.cc` のようにコンパイルオプションに `-std=c++11` を追加する．

C++11

```
#include <chrono>
using namespace std;
using namespace std::chrono;
  auto t0 = high_resolution_clock::now();
  solve(E);
  auto t1 = high_resolution_clock::now();
  cout << duration_cast<duration<double>>(t1-t0).count()
       << " seconds\n";
```

いずれの場合も，様々な理由で実行時間は試行毎に変化しうることに注意．

2.4 計算量の節約

問題	**Square Route**⋆	(模擬国内予選 2007)

縦横 1500 本程度の道路がある街で，正方形の数を数えてほしい．http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=2015

考察課題: 手法の説明と工夫した点，オーダを説明せよ．(考察が十分かどうか，なるべく授業中に，確認を受けること)

全ての 4 点を列挙して正方形になっているかどうかを試すと，$O(N^2M^2)$ となって間に合わない．様々な工夫の余地がある．ヒント:

問題	ほそながいところ ⋆	(夏合宿 2012)

馬車 n 台の出発時刻を調整して，条件を満たしながら全ての馬車がゴール地点に到着するまでにかかる時間の最小値を求める．

補足: この問題は難しいので，全部でのど可能性を試せばよいか，どのようにフィルタするか，解を求めるにはどのような関数を作成すれば良いか，方針をなるべく具体的に記述すれば評価する

http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=2427

2.5 今週の課題

問題を解いて，AOJ で Accepted を得たことを確認して，各問題指定の考察課題もしくは指定がない問題については感想 (学んだことなど) とともに，ITC-LMS に提出する．コメントの記述は，`#if 0` と`#endif` で囲むことが簡便である．

C++

```
1 // minmaxsum.cc
2 #if 0
3   所要時間は..であった. ...を復習した.
4   ...
5 #endif
6 int main() {
7 }
```

扱う問題は以下のいずれかで合格:(言語指定なし)

- 問題 “Tax Rate Changed” と “Space Coconut Grab” の両方を解く。Square Route の方針を検討する。
- Square Route(⋆) を解く。

提出先: ITC-LMS
提出の注意:

- 問題毎にテキストファイルを，提出する．内容はソースコードとコメントの文章とする．(プレーンテキストであること．つまり，PDF, Microsoft Word, rtf などは避けること)．グラフを作成した場合は，別ファイルで提出する．
  ⋆ つきの問題はオプションである．なお ⋆ つきの問題に (教員の基準で) 完全な回答の提出を行った場合は，⋆ なしの問題の提出を免除する．
- 提出物は履修者全体に公開される． この授業では，他者のソースコードを読んで勉強することを推奨するが，課題を解く上で参考にした場合は，何をどのように参加したかを明記すること．
- 締切: 授業終了時，次回授業の前日，学期終了時のそれぞれで最善のものを提出する

2.A 型パラメータ ⋆

(発展) 型を取り替えながら似たような処理を行いたい場合がある．C++ では `template` 関数という仕組みを用いることで，型以外の共通部分を関数としてまとめることができる．

C++

```
1  template <class T>
2  void show() { // 型 T の情報を表示
3    cout << sizeof(T)
4         << ' ' << numeric_limits<T>::digits
5         << ' ' << numeric_limits<T>::digits10
6         << ' ' << numeric_limits<T>::min()
7         << ' ' << numeric_limits<T>::max()
8         << endl;
9  }
10 int main() {
11   show<int>(); // T=int として show を呼ぶ
12   show<long>();// T=long として show を呼ぶ
13   show<long long>();
14   show<float>();
15   show<double>();
16   show<long double>();
17 }
```