アルゴリズム講座

第1回 〜導入・全探索編〜

2020/05/02
LeadHACK
@utsubo_21 高澤 栄一

Agenda

- Introduction (10分)
  - はじめに
  - 本講座の概要
- Lesson 1 (60分)
  - アルゴリズムの評価の仕方
  - 全探索
- 時間が余ったら実際に実装

Introduction

まず，はじめに

- アルゴリズムとデータ構造を学ぶ意義
  - ってなんだと思いますか？

はじめに

- 想定対象者
  - プログラミングの授業を受けた or 入門本は読んだ
    - 具体的には，配列を知っていれば大丈夫です
    - バブルソートを知っていれば完璧

はじめに

- 本講座で学ぶこと
  - 基本的なアルゴリズムとデータ構造の知識
    - どこで役に立っているのか・どのように実装するのかを意識して話す予定です
    - 一方で，あまり実装に寄りすぎず，アカデミックな内容も知る限りの話はします
    - AtCoder緑以上の人は本講座で学べることはないかも

講座概要

- 概要
  - 全10回位で基本的なアルゴリズムを網羅的に紹介
    - 完走出来るように頑張りましょう（頑張ります）

- 講座時間
  - 基本的に座学 1時間
  - 課題を出すので実装 2,30分（任意）
    - 講座終了後もslack等で随時質問を受け付けます

講座概要

- 下記のアルゴリズム・データ構造を10回程度に分けて１つずつ解説
  - 全探索
  - 二分探索
  - 深さ優先探索
  - 幅優先探索
  - 動的計画法
  - 累積和
  - グラフ・木
  - ダイクストラ法
  - ワーシャルフロイド法
  - クラスカル法
  - Union-Find

参考：レッドコーダーが教える、競プロ・AtCoder上達のガイドライン【中級編：目指せ水色コーダー！】
https://qiita.com/e869120/items/eb50fdaece12be418faa

Lesson 1

今回の内容

- アルゴリズムの評価の仕方 (25分)
  - オーダー記法
  - 時間/空間計算量のトレードオフ
- 休憩 (n分)
- 全探索 (20分)
  - 最も基本的な全探索
  - 2^Nの組み合わせを全探索

アルゴリズムの評価の仕方

アルゴリズムの評価の仕方

- オーダー記法
- 時間と空間計算量のトレードオフ
- 実際の処理時間の見積もり

アルゴリズムの評価の仕方

- どんなアルゴリズムが優れていると思いますか？

アルゴリズムの評価の仕方

- どんなアルゴリズムが優れていると思いますか？

高速　　　　省記憶領域

一般的には大きく分けてこの2つで評価

アルゴリズムの評価の仕方

- どんなアルゴリズムが優れていると思いますか？

高速　　　　**省記憶領域**

一般的には大きく分けてこの2つで評価

では，これをどのように定量的に扱うか？

アルゴリズムの評価の仕方

- どんなアルゴリズムが優れていると思いますか？

高速

指標: ステップ数

省記憶領域

使用メモリ量

アルゴリズムの評価の仕方

- どんなアルゴリズムが優れていると思いますか？

高速	省記憶領域
指標: ステップ数	使用メモリ量
↓	↓
時間計算量	空間計算量

計算量：

データ規模(入力長)に対して，どの程度資源を使うかを示す指標

計算量とオーダー記法

- 実装方法や使用言語によって,
  ステップ数や使用メモリ量は変わるため,
  計算量を厳密に見積もるのは難しい
  → オーダー記法で大雑把に見積もる！

計算量とオーダー記法

- 実装方法や使用言語によって,
  ステップ数や使用メモリ量は変わるため,
  計算量を厳密に見積もるのは難しい
  → オーダー記法で大雑把に見積もる！

- オーダー記法にも幾つか種類があるが,
  最も使われる**O(ビッグオー)記法**だけ紹介
  → 例を通してオーダー記法を説明します！

実際に計算量を見積もろう！

Example.

```
sum = 0
for i in 0..N {
  for j in 0..N {
    sum += a[i][j]
    sum += b[i][j]
  }
}
for i in 0..N {
  sum -= a[i][i];
}
```

このアルゴリズムの

「ステップ数」

を考える

2つの2次元の行列の総和を求め，

その後，aの対角成分を引いているが特に意味はない

実際に計算量を見積もろう！

Example.

```
sum = 0
for i in 0..N {
  for j in 0..N {
    sum += a[i][j]
    sum += b[i][j]
  }
}
for i in 0..N {
  sum -= a[i][i];
}
```

ステップ数

2*N^2 + N

N^2回

N^2回

N回

実際に計算量を見積もろう！

Example.

```
sum = 0
for i in 0..N {
  for j in 0..N {
    sum += a[i][j]
    sum += b[i][j]
  }
}
for i in 0..N {
  sum -= a[i][i];
}
```

ステップ数

2*N^2 + N

N^2回

N^2回

N回

実際に計算量を見積もろう！

- **O記法**では入力長Nが非常に大きくなった時のステップ数を考えて…

① Nが大きい時，最高次数の項の影響が大きいので,
それ以外は無視

② 係数は無視

実際に計算量を見積もろう！

- **O記法**では入力長Nが非常に大きくなった時のステップ数を考えて…

① Nが大きい時，最高次数の項の影響が大きいので，
それ以外は無視

$$2*N^2 + N \quad \rightarrow \quad 2*N^2$$

② 係数は無視

実際に計算量を見積もろう！

- **O記法**では入力長Nが非常に大きくなった時のステップ数を考えて…

① Nが大きい時，最高次数の項の影響が大きいので,
それ以外は無視

2*N^2 + N → 2*N^2

② 係数は無視

2*N^2 → N^2

実際に計算量を見積もろう！

- **O記法**では入力長Nが非常に大きくなった時のステップ数を考えて…

① Nが大きい時，最高次数の項の影響が大きいので，それ以外は無視

2*N^2 + N → 2*N^2

② 係数は無視

2*N^2 → N^2

「このアルゴリズムの計算量は O(N^2)」のような言い方をする

オーダー記法

- もう少し，**O記法**の練習を

100*n^4 + n + 100	= O(n^4)
n + log(n) + 10	= O(n)
n*log(n) + m	= O(n*log(n) + m)
V + E + V*log(V) + E*log(V)	= O((V + E)*log(V))
1024	= O(1)

Tips: オーダー記法

- O (ビッグオー) の定義

T(n) = O(f(n))
⇔
ある実数cと自然数n_0 が存在して，全てのn(≧n_0)に対して
T(n) ≦ cf(n)が成り立つ

最初の例で言えば,

T(n) = 2*n^2 + n = O(n^2)
f(n) = n^2

- **前ページの①, ② は，T(n)からf(n)へ上記の性質を満たしたまま変形**
- **定義的には，2*n^2 + n = O(n^3) でも間違いではない**

時間/空間計算量のトレードオフ

- 問題
  - M以下の整数Kが与えられるので，それが素数か判定せよ.
- 解法①
  - 1 ~ K-1までの数で割り切れるか試す
- 解法②
  - あらかじめMまでの各整数について，素数か否かを求め，配列aに入れておき，a[K]を答える

時間/空間計算量のトレードオフ

- 解法①
  - 1 ~ K-1までの数で割り切れるか試す

時間計算量：O(K)	**時間大**	**：K個の数値を試すため**
空間計算量：O(1)	**メモリ小**	**：1つずつ試すため**

時間/空間計算量のトレードオフ

- 解法①
  - 1 ~ K-1までの数で割り切れるか試す

    時間計算量：O(K)　時間大　：K個の数値を試すため

    空間計算量：O(1)　メモリ小　：1つずつ試すため

- 解法②
  - あらかじめMまでの各整数について,
    素数か否かを求めて配列aに入れておき，a[K]を答える

    時間計算量：O(1)　時間小　：配列アクセスのみ

    空間計算量：O(M)　メモリ大　：最大

時間/空間計算量のトレードオフ

- まとめると,
  - 高速にしようと思うと，メモリが必要
  - メモリを節約しようと思うと，速度が犠牲に

→ 時間計算量と空間計算量がトレードオフの関係に

実際の処理時間を見積もる

- ステップ数と計算時間の関係
  - 高速な言語の場合，おおよそ1億ステップで1,2秒程度
    - 処理が単純の場合，10倍くらいになることも
    - 例えば，O(n^2) の処理は，n=10000であれば1秒位
- 処理時間を見積もるメリット
  - バグや無限ループで応答がないのか，
    ただ処理に時間が掛かっているだけなのか判断できる

参考：実践・最強最速のアルゴリズム勉強会 第三回講義資料(ワークスアプリケーションズ & AtCoder)

https://www.slideshare.net/chokudai/wap-atcoder3

まとめ

- 頭の片隅に入れておいてほしいこと
  - 計算量とオーダー記法
    - 定数倍と最大次数の項以外は無視
  - 時間と空間計算量のトレードオフの関係
    - 速いほどメモリを食う傾向にある
  - 実際の処理時間の見積もり方
    - 1億回演算して1秒位

全探索

講座の具体的な内容

- 下記のアルゴリズム・データ構造を10回程度に分けて１つずつ解説

Lesson1
- 全探索
- 二分探索
- 深さ優先探索
- 幅優先探索
- 動的計画法
- 累積和
- グラフ・木
- ダイクストラ法
- ワーシャルフロイド法
- クラスカル法
- Union-Find

参考：レッドコーダーが教える、競プロ・AtCoder上達のガイドライン【中級編：目指せ水色コーダー！】
https://qiita.com/e869120/items/eb50fdaece12be418faa

全探索って？

- 全部のパターンをしらみつぶしに調べる方法
  - 一見，パワープレイで頭が悪そうだけど強力
  - 一般的な開発は殆どこれで済んでしまう (済まそうとする)
  - アルゴリズム系の研究でもファーストアプローチは基本的に全探索

- 全アルゴリズムの基礎 かつ 最重要！

全探索って？

- 全部のパターンをしらみつぶしに調べる方法
  - 〇〇なパターンが何通りあるか答えなさい
    - 全パターン試し，条件を満たすパターンをカウント
  - 〇〇なパターンが存在するか答えなさい
    - 全パターン試し，条件を満たすパターンが存在するか調べる
  - 〇〇するとき，最大値を求めなさい
    - 全パターン試し，一番大きいものを求める

- とにかく，全パターンを列挙することが重要！

参考：実践・最強最速のアルゴリズム勉強会　第一回　講義資料(ワークスアプリケーションズ & AtCoder)
https://www.slideshare.net/chokudai/wap-atcoder1

最も基本的な全探索

- 練習ついでに１問
  - M以下のN個の整数a[i]が与えれるので,
    その中に素数が幾つあるか？

どうすれば良いか少し考えてみましょう
どのように解くか？計算量は幾つか？

最も基本的な全探索

- 練習ついでに 1 問
  - M以下のN個の整数a[i]が与えれるので,
    その中に素数が幾つあるか？

```
for (i = 0; i < N; i++) {
  if (IsPrime(a[i]) {
    count += 1;
  }
}
```

forループで各整数が素数か全部見ていけば良い

計算量は，O(N * M)

では，こういう問題は？

- 部分和問題
  - N個の自然数a[i]と，自然数Wが与えられる．
    a[i]の中から何個か選んで総和をWにできるか？

では，こういう問題は？

- 部分和問題
  - N個の自然数a[i]と，自然数Wが与えられる.
    a[i]の中から何個か選んで総和をWにできるか？

例.

W : 75

a[i] : 3 12 29 43 69

では，こういう問題は？

- 部分和問題
  - N個の自然数a[i]と，自然数Wが与えられる.

    a[i]の中から何個か選んで総和をWにできるか？

例.

W : 75

a[i] : 3 12 29 43 69

75(=3+29+43) なので,Yes.

では，こういう問題は？

- 部分和問題
  - N個の自然数a[i]と，自然数Wが与えられる．
    a[i]の中から何個か選んで総和をWにできるか？

例えば，N=3であれば，

各i使うかどうかの

3重ループで解けそうだが…

```
for (use0 = 0; use0 < 2; use0++) {
  for (use1 = 0; use1 < 2; use1++) {
    for (use2 = 0; use2 < 2; use2++) {
      sum = 0;
      if (use0 == 1) sum += a[0];
      if (use1 == 1) sum += a[1];
      if (use2 == 1) sum += a[2];
      if (sum == W) return true;
    }
  }
}
```

では，こういう問題は？

- 部分和問題
  - N個の自然数a[i]と，自然数Wが与えられる．
    a[i]の中から何個か選んで総和をWにできるか？

N重forループを書けば解けるが…

Nは固定値ではない…

部分和問題

- 全ての部分集合を列挙したい

a[i] :	3	12	29	43	69
					69
				43	
			⋮		
			29	43	69
		12			
			⋮		

部分和問題

- 全ての部分集合を列挙したい

a[i] :	3	12	29	43	69	要素がある箇所にbitを立てた2進数
						00000
					69	00001
				43		00010
			⋮			⋮
			29	43	69	00111
		12				01000
			⋮			⋮

部分和問題

- 全ての部分集合を列挙したい

通し番号になっていて

forループで回せそう

a[i] :	3	12	29	43	69	要素がある箇所にbitを立てた2進数	10進数表記
						00000	0
					69	00001	1
				43		00010	2
		⋮				⋮	
			29	43	69	00111	7
		12				01000	8
		⋮				⋮	

部分和問題

- 部分集合の全列挙はforループできそう
  - どれくらい回せば良い？
    - 0 ~ 2^N-1まで（0は空集合，2^N-1は全要素）

```
for (set = 0; set < (1<<N); set++)
```

set が1つの組み合わせを表現している

部分和問題

- for文の中身を書いていこう
  - forループで回っている変数set から組み合わせを得る

部分和問題

- for文の中身を書いていこう
  - forループで回っている変数set から組み合わせを得る

*** set = 7 から組み合わせを得る場合**

10進数

表記

7

部分和問題

- for文の中身を書いていこう
  - forループで回っている変数set から組み合わせを得る

*** set = 7 から組み合わせを得る場合**

要素がある箇所にbitを立てた2進数		10進数表記
00111	←	**7**

部分和問題

- for文の中身を書いていこう
  - forループで回っている変数set から組み合わせを得る

*** set = 7 から組み合わせを得る場合**

a[i] :	**3**	**12**	**29**	**43**	**69**		**要素がある箇所にbitを立てた2進数**	**10進数表記**
			29	**43**	**69**	⇐	**00111**	⇐ **7**

部分和問題

- for文の中身を書いていこう
  - forループで回っている変数set から組み合わせを得る

*** set = 7 から組み合わせを得る場合**

a[i] :	**3**	**12**	**29**	**43**	**69**	**要素がある箇所にbitを立てた2進数**	**10進数表記**
			29	**43**	**69**	**00111**	**7**

2進数表現でbit

が立っている位置

がsetが含む要素

部分和問題

setで表現した組み合わせに

右からi番目の要素が含まれるか調べる条件

```
if (set & (1 << i))
```

例えば，set = 7なら…

2番目の要素があるか

```
   set(7)    00111
&  1 << 2    00100
--------------------
             00100
```

3番目の要素があるか

```
   set(7)    00111
&  1 << 3    01000
--------------------
             00000
```

部分和問題

setで表現した組み合わせに

右からi番目の要素が含まれるか調べる条件

```
if (set & (1 << i))
```

例えば，set = 7なら…

2番目の要素があるか

```
    set(7)    00111
&   1 << 2    00100
---------------------
              00100
```

True

3番目の要素があるか

```
    set(7)    00111
&   1 << 3    01000
---------------------
              00000
```

False

部分和問題 setで表現した組み合わせに

右からi番目の要素が含まれるか調べる条件

```
if (set & (1 << i))
```

例えば，set = 7なら…

i	4	3	2	1	0
1 << i	10000	01000	00100	00010	00001
set (2進数)	00111	00111	00111	00111	00111
set & (1 << i)	00000	00000	00100	00010	00001
右からi番目が含まれるか	false	false	true	true	true

部分和問題

- for文の中身を書いていこう

```
// 部分和を計算
sum = 0;
// 各要素が set に含まれるか調べる
for (i = 0; i < N; i++) {
  // 右からi番目が存在
  if (set & (1 << i)) {
    sum += a[i];
  }
}
```

部分和問題

- 組み合わせると…

```
for (set = 0; set < (1<<N); set++) {
  // 部分和を計算
  sum = 0;
  // 各要素が set に含まれるか調べる
  for (i = 0; i < N; i++) {
    // 右からi番目が存在
    if (set & (1 << i)) {
      sum += a[i];
    }
  }
  if (sum == W) return true;
}
```

では，こういう問題は？

- 部分和をforループで全列挙することで解けた
  - 部分集合（2^Nの組み合わせ）を全列挙できる！
  - このような実装テクニックを競プロ界隈では
    bit全探索と呼んでいます

では，こういう問題は？

- 部分和をforループで全列挙することで解けた
  - 部分集合（2^Nの組み合わせ）を全列挙できる！
  - このような実装テクニックを競プロ界隈では bit全探索と呼んでいます
- 注意点
  - 計算量はO(2^N)でN=27で1億を超えてしまう
  - O(2^N)のように肩にNが乗ると，Nが小さい場合しか実用的な時間で実行できない

まとめ

- 全探索
  - 全探索は全ての基本
- 2^N組み合わせの全列挙
  - bit全探索
  - O(2^N)のアルゴリズムはNが小さい時には実用的

次回は？

- 二分探索 / 二分法
  - 次回はアルゴリズムらしいアルゴリズムなので,
    もう少し面白いと思います！（多分）

自由課題

- C - Half and Half：https://atcoder.jp/contests/abc095/tasks/arc096_a
    - 基本的なforループでの探索です
        - 工夫して全探索する部分を減らしましょう
- C - HonestOrUnkind2： https://atcoder.jp/contests/abc147/tasks/abc147_c
    - bit全探索です
        - 問題が少し複雑ですがやることは全探索です