

スケジュール 後半 (予定)

- 8 離散確率論：確率的離散システムの解析 (12/13)
- * 中間試験 (12/20)
- 9 離散確率論：乱択データ構造とアルゴリズム (基礎) (1/10)
- 10 離散確率論：乱択データ構造とアルゴリズム (発展) (1/17)
- 11 離散確率論：マルコフ連鎖 (基礎) (1/24)
- 12 離散確率論：マルコフ連鎖 (発展) (1/31)
- 13 離散確率論：マッシュアップ (2/7)
- * 期末試験 (2/14?)

注意：予定の変更もありうる

乱択アルゴリズム

乱択アルゴリズムとは？

乱数を用いる (あるいは、用いてもよい) アルゴリズムのこと

確率的アルゴリズム, 乱数使用アルゴリズムとも呼ばれる

なぜ乱数を用いるのか？

- ▶ アルゴリズムを設計しやすくなる
- ▶ アルゴリズムを解析しやすくなる
- ▶ 乱数を使わないとできないことが、乱数を使うとできる

前進問題

前進問題

設定

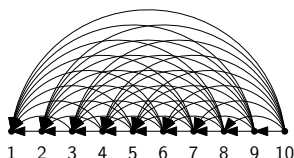
- ▶ 頂点集合を $\{1, 2, \dots, n\}$ とする有向グラフ ($n \geq 2$)
- ▶ 大きな数から小さな数へ向かう辺が必ず存在

行うこと

- ▶ 頂点 n から始めて、辺をたどることで頂点 1 に到達

問題

- ▶ 辺をいくつたどれば頂点 1 に到達できるか？



スケジュール 前半

- 1 数え上げの基礎：二項係数と二項定理 (10/4)
- * 休講 (10/11)
- 2 数え上げの基礎：漸化式の立て方 (10/18)
- 3 数え上げの基礎：漸化式の解き方 (基礎) (10/25)
- * 休講 (11/1)
- 4 数え上げの基礎：漸化式の解き方 (発展) (11/8)
- 5 離散代数：対称群と置換群 (11/15)
- * 休講 (11/22)
- 6 離散代数：有限群 (11/29)
- 7 離散代数：有限群の応用 (12/6)

今日の目標

今日の目標

典型的な乱択アルゴリズムの解析ができるようになる

- ▶ 前進問題
- ▶ 乱択クイックソート

前進問題

目次

- 1 前進問題
- 2 乱択クイックソート
- 3 今日のまとめ

前進問題

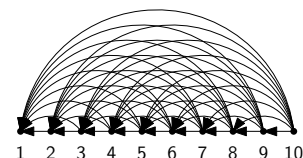
単純なアルゴリズム 1 (乱数を使わない)

- 1 たどる辺を任意に選び、辺の先に移動する
- 2 移動先から出る辺がなければ終了, そうでなければ 1 に戻る

たどる辺の数

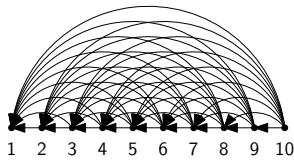
- ▶ 最悪の場合： $n - 1$ 個
- ▶ (最善の場合：1 個)

最悪計算量の意味では、よくないアルゴリズム



- 1 たどる辺を一様分布に従って選び、辺の先に移動する
- 2 移動先から出る辺がなければ終了、そうでなければ 1 に戻る

一様分布に従って選ぶ \equiv 出る辺が k 個ある場合、それぞれを確率 $1/k$ で選ぶ



証明 (1)

- ▶ 任意の $k \in \{1, 2, \dots, n\}$ に対して、次の確率変数 R_k を定義
 $R_k =$ 頂点 k から始めて、頂点 1 への到達までにたどる辺数
- ▶ このとき、 $E[R_1] = 0$ で、 $k \geq 2$ のとき、

$$\begin{aligned} E[R_k] &= \sum_{i=1}^{k-1} E[R_k | \text{頂点 } k \text{ から頂点 } i \text{ に向かう辺を選ぶ}] \\ &\quad \cdot \Pr(\text{頂点 } k \text{ から頂点 } i \text{ に向かう辺を選ぶ}) \\ &= \sum_{i=1}^{k-1} (E[1 + R_i]) \cdot \frac{1}{k-1} = \sum_{i=1}^{k-1} (1 + E[R_i]) \cdot \frac{1}{k-1} \\ &= 1 + \frac{1}{k-1} \sum_{i=1}^{k-1} E[R_i] \end{aligned}$$

- ▶ この再帰式を解きたい

証明 (2)

- ▶ $k \geq 2$ のとき、両辺を $k-1$ 倍すると

$$(k-1)E[R_k] = k-1 + \sum_{i=1}^{k-1} E[R_i]$$

- ▶ よって、 $k \geq 3$ のとき

$$(k-2)E[R_{k-1}] = k-2 + \sum_{i=1}^{k-2} E[R_i]$$

- ▶ 上の式から下の式を引くと、 $k \geq 3$ のとき

$$\begin{aligned} (k-1)E[R_k] - (k-2)E[R_{k-1}] &= 1 + E[R_{k-1}] \\ (k-1)E[R_k] &= 1 + (k-1)E[R_{k-1}] \\ E[R_k] &= \frac{1}{k-1} + E[R_{k-1}] \end{aligned}$$

期待値が分かるとなぜよいか？

- ▶ たどる辺数の期待値が分かったからといって、アルゴリズムがそれだけの辺数しかたどらないとは限らない (乱数を使っているから)
- ▶ しかし、マルコフの不等式から

$$\begin{aligned} \Pr(R_n \geq 2H_{n-1}) &\leq \frac{E[R_n]}{2H_{n-1}} \\ &= \frac{H_{n-1}}{2H_{n-1}} = \frac{1}{2} \end{aligned}$$

- ▶ つまり、 $\Pr(R_n < 2H_{n-1}) \geq 1 - \frac{1}{2} = \frac{1}{2}$
- ▶ $\frac{1}{2}$ 以上の確率で、たどる辺数は少ない ($2H_{n-1}$ 未満)

しっかりとした確率を導出するために、チェルノフ上界の技法を使う

証明すること

単純なアルゴリズム 2 がたどる辺の数の期待値は H_{n-1}

復習: H_{n-1} は $n-1$ 次調和数であり、

$$H_{n-1} = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n-1}$$

事実として、 $H_{n-1} = \ln n + O(1)$ が成り立つ

- ▶ つまり、単純なアルゴリズム 2 がたどる辺の数の期待値は $O(\log n)$

解きたい再帰式: 再掲

解きたい再帰式

$$E[R_k] = \begin{cases} 0 & (k=1) \\ 1 + \frac{1}{k-1} \sum_{i=1}^{k-1} E[R_i] & (k \geq 2) \end{cases}$$

$k \geq 2$ のとき、両辺を $k-1$ 倍すると

$$(k-1)E[R_k] = k-1 + \sum_{i=1}^{k-1} E[R_i]$$

証明 (3)

- ▶ したがって、 $k \geq 3$ のとき

$$\begin{aligned} E[R_k] &= \frac{1}{k-1} + E[R_{k-1}] \\ &= \frac{1}{k-1} + \frac{1}{k-2} + E[R_{k-2}] \\ &= \dots \\ &= \frac{1}{k-1} + \frac{1}{k-2} + \dots + \frac{1}{2} + \underbrace{E[R_2]}_{=1} \\ &= H_{k-1} \end{aligned}$$

- ▶ 特に、 $n \geq 2$ に対して、

$$E[R_n] = H_{n-1}$$

□

前進問題: チェルノフ上界の技法

- ▶ R_n の代わりに 2^{R_n} を考えてみる

証明したいこと

任意の $k = 1, 2, \dots, n$ に対して

$$E[2^{R_k}] = k$$

- ▶ これが正しいとすると、

$$\begin{aligned} \Pr(R_n \geq 2 \log_2 n) &= \Pr(2^{R_n} \geq 2^{2 \log_2 n}) \\ &= \Pr(2^{R_n} \geq n^2) \\ &\leq \frac{E[2^{R_n}]}{n^2} = \frac{1}{n} \end{aligned}$$

- ▶ つまり、 $1 - \frac{1}{n}$ 以上の確率でたどる辺数は少ない ($2 \log_2 n$ 未満)

$$\Pr(R_n < 2 \log_2 n) \geq 1 - \frac{1}{n}$$

先ほどと同様な手順を進める

- ▶ $E[2^{R_1}] = 2^0 = 1$, $E[2^{R_2}] = 2^1 = 2$ で, $k \geq 2$ のとき,

$$\begin{aligned} E[2^{R_k}] &= \sum_{i=1}^{k-1} E[2^{R_k} | \text{頂点 } k \text{ から頂点 } i \text{ に向かう辺を選ぶ}] \\ &\quad \cdot \Pr(\text{頂点 } k \text{ から頂点 } i \text{ に向かう辺を選ぶ}) \\ &= \sum_{i=1}^{k-1} E[2^{1+R_i}] \cdot \frac{1}{k-1} = \sum_{i=1}^{k-1} E[2 \cdot 2^{R_i}] \cdot \frac{1}{k-1} \\ &= \sum_{i=1}^{k-1} 2 E[2^{R_i}] \cdot \frac{1}{k-1} = \frac{2}{k-1} \sum_{i=1}^{k-1} E[2^{R_i}] \end{aligned}$$

- ▶ この再帰式を解きたい

- ▶ $k \geq 2$ のとき, 両辺を $k-1$ 倍すると

$$(k-1)E[2^{R_k}] = 2 \sum_{i=1}^{k-1} E[2^{R_i}]$$

- ▶ よって, $k \geq 3$ のとき

$$(k-2)E[2^{R_{k-1}}] = 2 \sum_{i=1}^{k-2} E[2^{R_i}]$$

- ▶ 上の式から下の式を引くと, $k \geq 3$ のとき

$$\begin{aligned} (k-1)E[2^{R_k}] - (k-2)E[2^{R_{k-1}}] &= 2E[2^{R_{k-1}}] \\ E[2^{R_k}] &= \frac{k}{k-1} E[2^{R_{k-1}}] \end{aligned}$$

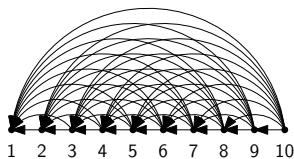
乱数を使わないアルゴリズム

- ▶ 最悪時：たどる辺数 = $n-1$

乱択アルゴリズム

- ▶ 期待値：たどる辺数 = $H_{n-1} (= \ln n + O(1))$
- ▶ 高確率：たどる辺数 = $O(\log n)$

∴ 乱数を使うことで, 問題を高速に解けた



ソーティング (整列問題) とは？

- ▶ 入力：異なる n 個の数から成る配列 $A = (a_1, a_2, \dots, a_n)$ (配列)
- ▶ 出力： A の並べ替え $A' = (a'_1, a'_2, \dots, a'_n)$ で, $a'_1 < a'_2 < \dots < a'_n$ を満たすもの

例： $A = (8, 3, 5, 1, 7, 9, 2, 4) \rightsquigarrow A' = (1, 2, 3, 4, 5, 7, 8, 9)$

アルゴリズムにおける基本的な問題

解きたい再帰式

$$E[2^{R_k}] = \begin{cases} 1 & (k=1) \\ \frac{2}{k-1} \sum_{i=1}^{k-1} E[2^{R_i}] & (k \geq 2) \end{cases}$$

- ▶ したがって, $k \geq 3$ のとき

$$\begin{aligned} E[2^{R_k}] &= \frac{k}{k-1} E[2^{R_{k-1}}] \\ &= \frac{k}{k-1} \cdot \frac{k-1}{k-2} E[2^{R_{k-2}}] \\ &= \dots \\ &= \frac{k}{k-1} \cdot \frac{k-1}{k-2} \cdot \dots \cdot \frac{3}{2} E[2^{R_2}] \\ &= k \end{aligned}$$

□

① 前進問題

② 乱択クイックソート

③ 今日のまとめ

クイックソート

再帰によってソーティングを行うアルゴリズム (の1つ)

- 1 A から要素を1つ選択 (その要素を **ピボット** と呼ぶ)
- 2 A を3つの部分に分割
 - ▶ A_1 : ピボットよりも小さい要素から成る配列
 - ▶ p : ピボット
 - ▶ A_2 : ピボットよりも大きい要素から成る配列
- 3 A_1 と A_2 を再帰的に整列 (結果をそれぞれ A'_1, A'_2 とする)
- 4 A'_1 と p と A'_2 をこの順に連結して出力

- ▶ アルゴリズムの正当性は直ちに分かる
- ▶ ピボットの選択法, A_1, A_2 の作成法によって, 細かな実装が変わる

ピボットの選択法, A_1, A_2 の作成法によって, 細かな実装が変わる

よく使われるピボット選択法

- ▶ 配列の先頭の要素をピボットとする
- ▶ 配列の先頭の3要素の中央値をピボットとする
- ▶ 配列の中のランダムな要素をピボットとする (乱択アルゴリズム) (各要素が選択される確率は同一 (一様分布に従う標本抽出))

乱択クイックソート (Ruby)

```

1: def quicksort(a)
2:   return nil.to_a if a.length == 0
3:   p = a.sample()
4:   a.delete(p)
4': a1 = Array.new(); a2 = Array.new()
5:   a.each { |e|
6:     print "G"
7:     e < p ? a1 << e : a2 << e
8:   }
9:   return quicksort(a1) + [p] + quicksort(a2)
10: end

```

X_A = 入力 A に対する乱択クイックソートの比較回数 (確率変数)
 x_n = 入力サイズを n としたときの, 最悪時期待比較回数
 = $\max\{E[X_A] \mid |A| = n\}$ (実数)

目標：次を証明すること

x_n が小さいこと (具体的には, $x_n = O(n \log n)$ となること)

そのために x_n が満たす漸化式を導出する

- ▶ $x_0 = 0$ (要素数 0 の入力に対して, 比較は行わない)

$n \geq 1$ のとき, $x_n = E[X_A]$ となるような入力 A を考えてみると

$$x_n = \sum_{i=1}^n E[X_A \mid a_i \text{ がピボット}] \frac{1}{n}$$

(ただし, a_i は A の中で i 番目に小さい要素)

ここで, a_i がピボットであるとき

- ▶ $|A_1| = i - 1, |A_2| = n - i$
- ▶ $\therefore E[X_{A_1}] \leq x_{i-1}$ かつ $E[X_{A_2}] \leq x_{n-i}$

したがって, $n \geq 1$ のとき

$$x_n \leq \sum_{i=1}^n (n - 1 + x_{i-1} + x_{n-i}) \frac{1}{n} = n - 1 + \sum_{i=0}^{n-1} \frac{2}{n} x_i$$

乱択クイックソート (適当な疑似コード)

```

1: def quicksort(A) # A: array of distinct numbers
2:   return nil if length(A) == 0
3:   p = a number in A chosen uniformly at random
4:   delete p from A
5:   foreach e in A {
6:     print "G"
7:     if e < p then add e to A1 else add e to A2
8:   }
9:   return quicksort(A1) + p + quicksort(A2)
10: end

```

評価尺度として, 以下のものがよく用いられる

- ▶ 比較回数: 2 要素の比較を行った回数
- ▶ 移動回数: 要素を移動した回数
- ▶ 領域量: 入力配列以外に用いた変数の数

ここでは, 比較回数に注目 (比較回数 = 出力された G の個数)

- ▶ 乱択クイックソートにおいて, 比較回数は使用される乱数によって変わる (つまり, 確率変数)
 X_A = 入力 A に対する乱択クイックソートの比較回数 (確率変数)

$n \geq 1$ のとき, $x_n = E[X_A]$ となるような入力 A を考えてみると

$$\begin{aligned}
 x_n &= E[X_A] \\
 &= \sum_{i=1}^n E[X_A \mid a_i \text{ がピボット}] \Pr[a_i \text{ がピボット}] \\
 &\quad (\text{ただし, } a_i \text{ は } A \text{ の中で } i \text{ 番目に小さい要素}) \\
 &= \sum_{i=1}^n E[X_A \mid a_i \text{ がピボット}] \frac{1}{n}
 \end{aligned}$$

 x_n に関して得られた漸化式

$$\begin{aligned}
 x_0 &= 0 \\
 x_n &\leq n - 1 + \sum_{i=0}^{n-1} \frac{2}{n} x_i \quad (n \geq 1)
 \end{aligned}$$

ここで, 次の漸化式を満たす数列 $\{t_n\}_{n \geq 0}$ を考える

$$\begin{aligned}
 t_0 &= 0 \\
 t_n &= n - 1 + \sum_{i=0}^{n-1} \frac{2}{n} t_i \quad (n \geq 1)
 \end{aligned}$$

- ▶ このとき, 任意の $n \geq 0$ に対して次が成り立つ (演習問題)

$$x_n \leq t_n$$

- ▶ つまり, t_n の上界が分かれば, x_n の上界となる

漸化式を解く (1)

$$t_n = n - 1 + \sum_{i=0}^{n-1} \frac{2}{n} t_i \quad (n \geq 1)$$

この式において、添え字をずらしたものを考えると

$$t_{n+1} = n + \sum_{i=0}^n \frac{2}{n+1} t_i \quad (n \geq 0)$$

これら2つの式を変形すると

$$nt_n = (n-1)n + \sum_{i=0}^{n-1} 2t_i \quad (n \geq 1)$$

$$(n+1)t_{n+1} = n(n+1) + \sum_{i=0}^n 2t_i \quad (n \geq 0)$$

下から上を引くと、 $n \geq 1$ のとき、

$$(n+1)t_{n+1} - nt_n = 2n + 2t_n$$

漸化式を解く (3)

$n \geq 2$ のとき、

$$\begin{aligned} s_n &= \frac{2}{n+1} - \frac{1}{n} + s_{n-1} \\ &= \left(\frac{2}{n+1} - \frac{1}{n} \right) + \left(\frac{2}{n} - \frac{1}{n-1} \right) + s_{n-2} \\ &= \left(\frac{2}{n+1} - \frac{1}{n} \right) + \left(\frac{2}{n} - \frac{1}{n-1} \right) + \cdots + \left(\frac{2}{3} - \frac{1}{2} \right) + s_1 \\ &= \frac{2}{n+1} + \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{3} - \frac{1}{2} \\ &= \frac{1}{n+1} + H_{n+1} - \frac{1}{2} - 1 - \frac{1}{2} \\ &= H_{n+1} + \frac{1}{n+1} - 2 \end{aligned}$$

復習: $H_{n+1} = 1 + \frac{1}{2} + \cdots + \frac{1}{n+1}$ (調和数)

乱択クイックソートの解析: まとめ

ここまでで分かったこと

任意の $n \geq 0$ と、 $|A| = n$ であるような任意の入力 A に対して

$$E[X_A] \leq 2(n+1) \ln(n+1)$$

したがって、マルコフの不等式を適用してみると

$$\begin{aligned} \Pr[X_A \geq 4(n+1) \ln(n+1)] &\leq \frac{E[X_A]}{4(n+1) \ln(n+1)} \\ &\leq \frac{2(n+1) \ln(n+1)}{4(n+1) \ln(n+1)} = \frac{1}{2} \end{aligned}$$

- ▶ つまり、比較回数が $4(n+1) \ln(n+1)$ を超える確率は高くない
- ▶ 「チェルノフ上界の技法」を用いると、 $n \rightarrow \infty$ のとき、この確率が 0 に収束することを証明できる (ちよっと面倒で、他のアイデアも必要なので、省略)

今日の目標

今日の目標

典型的な乱択アルゴリズムの解析ができるようになる

- ▶ 前進問題
- ▶ 乱択クイックソート

漸化式を解く (2)

整理すると、 $n \geq 1$ のとき、

$$(n+1)t_{n+1} = 2n + (n+2)t_n$$

両辺を $2(n+1)(n+2)$ で割ると、 $n \geq 1$ のとき、

$$\frac{t_{n+1}}{2(n+2)} = \frac{n}{(n+1)(n+2)} + \frac{t_n}{2(n+1)}$$

ここで、 $s_n = \frac{t_n}{2(n+1)}$ と置くと、得られる漸化式は

$$s_0 = \frac{t_0}{2(0+1)} = 0$$

$$s_1 = \frac{t_1}{2(1+1)} = 0$$

$$s_{n+1} = \frac{2}{n+2} - \frac{1}{n+1} + s_n \quad (n \geq 1)$$

解けそうな形に近づいてきた

漸化式を解く (4)

したがって、 $n \geq 0$ に対して、

$$s_n = H_{n+1} + \frac{1}{n+1} - 2$$

したがって、 $n \geq 0$ に対して、

$$t_n = 2(n+1)s_n = 2(n+1)H_{n+1} + 2 - 4(n+1)$$

したがって、

$$x_n \leq t_n = 2(n+1)H_{n+1} + 2 - 4(n+1)$$

$H_{n+1} \leq 1 + \ln(n+1)$ なので (前回の演習問題)

$$\begin{aligned} x_n &\leq 2(n+1)(1 + \ln(n+1)) + 2 - 4(n+1) \\ &= 2(n+1) \ln(n+1) - 2n = O(n \log n) \end{aligned}$$

目次

- 1 前進問題
- 2 乱択クイックソート
- 3 今日のまとめ

残った時間の使い方

- ▶ 演習問題をやる
 - ▶ 相談推奨 (ひとりでやらない)
- ▶ 質問をする
 - ▶ 教員と TA は巡回
- ▶ 退室時、小さな紙に感想など書いて提出する ← 重要
 - ▶ 内容は何でも OK
 - ▶ 匿名で OK