実践的アルゴリズム
(2010年9月15日・於金沢大学)

上原隆平
uehara@jaist.ac.jp
http://www.jaist.ac.jp/~uehara/
Google に “上原　折り紙” でもOK!

1

困難な問題に対する挑戦

- (計算機で解きたい)困難な問題は
  - 解答の妥当性は、ある程度わかる
  - 可能な選択肢が爆発的に大きくて手に負えない
  - 理論的なモデルでは
    - 最悪の場合の計算量の下界を与える
    - 正確な解を求める
    - 入力サイズが大きくなったときの漸近的なふるまい

    を示すことが多い

もう少しなんとかしたい！！

2/40

困難な問題に対する挑戦

- (計算機で解きたい)困難な問題は
  - 理論的なモデルでは
    - 最悪の場合の計算量の下界を与える
    - 正確な解を求める
    - 入力サイズが大きくなったときの漸近的なふるまい

    を示すことが多い

もう少しなんとかしたい！！

- 最近のトレンド:
  - ランダム性を使って平均的なふるまいを考える　**乱択アルゴリズム**
  - 正解からの誤差を許した計算を考える　**近似アルゴリズム**
  - (指数時間かけてでも、ある程度の規模の問題を解く)

3/40

困難な問題に対する挑戦

- (計算機で解きたい)困難な問題は
  - 理論的なモデルでは
  - 最近のトレンド:
    - ランダム性を使った平均的なふるまい:**乱択アルゴリズム**
      - “ランダム性”を入力に仮定する
        - » …この場合はアルゴリズムは決定性でもよい
        - » 一般に与えられる入力の分布はわからないことが多い
      - アルゴリズムで“乱数”を使う
        - » “良い”疑似乱数を生成する方法はけっこう難しい
      - 効率(実行時間/使用メモリ)の期待値/最悪値などを解析する必要がある
    - 正解からの誤差を許した計算を考える:**近似アルゴリズム**
      - 近似率を理論的に保証することができる場合がある

4/40

確率解析…の前に

- アルゴリズムで“乱数”を使う
  - “良い”疑似乱数を生成する方法はけっこう難しい
    - 良い疑似乱数を使用することで、モンテカルロ法が1000倍速くなり、それまで2年かかっていた計算が1日で終わるようになった事例がある。
    - 疑似乱数の「良さ」とは？
      コンピュータ・サイエンスのバイブル
      “The Art of Computer Programming”, Vol. 2, by D.E. Knuth
      でも詳しく議論されているテーマ
  - **メルセンヌ・ツイスター(Mersenne twister)**
    - 松本眞、西村拓士による疑似乱数生成器
    - 公式サイトからダウンロードして使用可能
    - http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT/index-jp.html
    - 速い/コードが単純/性能が理論保証されている

5/40

確率解析の例: クーポンコレクター問題

- クーポンコレクター問題

> n 種類のクーポンが無限にある。毎回それぞれのクーポンを $1/n$ の確率で引き当てる。すべてのクーポンが集まるまでクーポンを引き続けるとすると、クーポンを何枚引かなければならないだろうか。

- 定理

> n 種類のクーポンが集まるまでにクーポンを引く枚数を$C(n)$とおくと、$C(n)$の期待値 $E(C(n))$ は次の式で表現される:
> $$E(C(n)) = O(n \log n)$$

6/40

確率解析の例: クーポンコレクター問題

– 定理

n 種類のクーポンが集まるまでにクーポンを引く枚数を$C(n)$とおくと、$C(n)$の期待値 $E(C(n))$ は次の式で表現される:

$$E(C(n)) = O(n \log n)$$

– おまけ: 上記の式は $E(C(n)) = n \log n + o(1)$ と書ける。

– 例:
- クーポンが10種類なら23強
- クーポンが100種類なら460強

余談: マクドナルドの101匹わんちゃんのコンプリートセットは定価5万円でした。

余談: オークションで7000円でした。

確率解析の例: クーポンコレクター問題

– 定理: n 種類のクーポンが集まるまでにクーポンを引く枚数を$C(n)$とおくと、$C(n)$の期待値 $E(C(n))$ は次の式で表現される: $E(C(n)) = O(n \log n)$

– 証明
- 現在 i 種類のクーポンを持っている状態を状態 i とする。
- 時刻 t に t 枚目のクーポンを引くとする。
- 時刻0は状態0で、時刻1は状態1である。
- 状態 i でクーポンを1枚引くと、
  - 新しいクーポンを引いて状態 i+1 に状態遷移する確率: $(n\text{-}i)/n$
  - すでに持っているクーポンを引いて状態が変わらない確率: i/n
- 新しい状態に遷移することを**成功**と考えると、これは幾何分散に関するよく知られた定理が使える。

確率解析の例: クーポンコレクター問題

– 幾何分散に関するよく知られた定理: 1回の試行で成功する確率が p である事象を繰り返し行うと、1回成功するまでの繰り返しの回数の期待値 $E(S(p))$ は次の式で表現される: $E(S(p)) = 1/p$

– 例:
- 成功確率が1/2なら繰り返しの回数の期待値は2回
- 成功確率が1/6なら繰り返しの回数の期待値は6回
- 成功確率が1/100なら繰り返しの回数の期待値は100回

– 証明
- 定義通り、
$$1 \cdot p + 2 \cdot (1-p) \cdot p + 3 \cdot (1-p)^2 p + \cdots + i \cdot (1-p)^{i-1} p + \cdots$$
を計算すればよい。

– 注意: あくまで期待値の話であって、博打には役立ちません

確率解析の例: クーポンコレクター問題

– 定理: n 種類のクーポンが集まるまでにクーポンを引く枚数を$C(n)$とおくと、$C(n)$の期待値 $E(C(n))$ は次の式で表現される: $E(C(n)) = O(n \log n)$

– 証明
- 状態 i でクーポンを1枚引くと、
  - 新しいクーポンを引いて状態 i+1 に状態遷移する確率: $(n\text{-}i)/n$
  - すでに持っているクーポンを引いて状態が変わらない確率: i/n
- 新しい状態に遷移することを**成功**と考えると、これは幾何分散に関するよく知られた定理が使える。
- よって状態 i から状態 i+1 に遷移するまでの時間の期待値＝状態 i の区間の長さの期待値＝$n/(n\text{-}i)$
- 期待値の線形性より、これらを i=0〜n-1まで足せばよい
（意外と重要: 期待値には線形性があり、そのまま足すことができる。）

確率解析の例: クーポンコレクター問題

– 定理: n 種類のクーポンが集まるまでにクーポンを引く枚数を$C(n)$とおくと、$C(n)$の期待値 $E(C(n))$ は次の式で表現される: $E(C(n)) = O(n \log n)$

– 証明
- したがって求める期待値は
$$\sum_{i=0}^{n-1} \frac{n}{n-i} = n \sum_{i=0}^{n-1} \frac{1}{n-i} = n \sum_{i=1}^{n} \frac{1}{i} = nH(n)$$
となる
- ここで $\sum_{i=1}^{n} \frac{1}{i} = H(n)$ は調和数 (Harmonic number)と呼ばれる数列で
$$H(n) = \log n + \gamma + \frac{1}{n} - \frac{1}{12n^2} + \frac{1}{120n^4} - \varepsilon,\ 0 < \varepsilon < \frac{1}{256n^6},\ \gamma = 0.5772156649\cdots$$
が知られている。よって定理を得る。

確率解析の例: クーポンコレクター問題

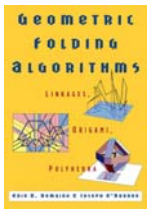
– 調和数についての参考文献
- 簡単に読めて楽しい（？）本

– 『数学ガール』結城浩、ソフトバンククリエイティブ、2007.
ハーモニックナンバーに関する章だけなら以下のWeb版でも読めます:
» テトラちゃんとハーモニック・ナンバー
http://www.hyuki.com/girl/harmonic.pdf
結城浩、2006年4月

– 『世界でもっとも奇妙な数学パズル』ジュリアン・ハヴィル著、松浦俊輔訳、青土社、2009.
第11章に載ってます。いろいろな「不思議」が載っている良書。

ちょっと休憩…

- レポート課題を配布する



乱択アルゴリズム(Randomized Algorithm)

– 乱数を用いたアルゴリズム
  - 乱数を用いることで計算の高速化/単純化を目指すアルゴリズム
  - ラスベガス型
    – いつでも正しい答えを返すことが保証されている。
    – 計算時間は何らかの確率分布に従う
  - モンテカルロ型
    – ときどき間違える。例えば Yes/No タイプの問題であれば
      » two-sided error; Yes を No という確率も No を Yes という確率も0ではない
      » one-sided error; 上記のどちらかのみ 0 でない
    – 普通は誤答を与える確率を小さくできる

乱択アルゴリズム(Randomized Algorithm)

– 乱数を用いたアルゴリズム
  - ラスベガス型:いつでも正しい解答を出力する
  - モンテカルロ型:一定の割合で誤答を出力する
  - 演習問題1:
    – アルゴリズムAは1回の実行時間が $t(n)$ 時間であり、そのとき3/4の確率で正解を出力する。これを3回実行して多数決をとる。このときの実行時間と正解を出力する確率を求めよ。
  - レポート問題1:
    – アルゴリズムBは1回の実行時間が $t_1(n)$ 時間である。このとき確率 p で正解を出力する。出力が正しいかどうかは別のアルゴリズム C で $t_2(n)$ 時間で確認することができる。正解が得られるまで B, C を繰り返し実行するアルゴリズムの実行時間と正解を出力する確率を求めよ。ここから何が言えるか。

確率解析の例(1): QuickSortの解析

– ソーティング問題
  Input: n 個のデータを記録した配列 a[n]
  Output: 以下の条件を満たす配列 a[n]
    a[1]<a[2]<…<a[n]
  ★話を単純にするため、a[i]=a[j], i≠j を満たすペアはないと仮定

– QuickSort は実用上、最速と言われることが多い
  - 典型的な分割統治法に基づくアルゴリズム
  - 都合良く分割されると $O(n \log n)$ 時間で計算が終わる
  - いつでも最悪の場合だと $O(n^2)$ かかる
  - …理論的な解析と速度保証はできるのか？

確率解析の例(1): QuickSortの解析

– QuickSortのおさらい
  - qsort(a,1,n) を呼び出す
  - qsort(a, i, j) が呼び出されると、
    – pivot a[m] を(ランダムに)選ぶ
    – a を a[m] を基準に「前半」と「後半」に分ける。つまり
      i≦ i' < m なら a[i']<a[m]
      m< j' < j なら a[j']>a[m]
      を満たすように並べ替える
    – qsort(a, i, i'), a[m], qsort(a, j', j) がソート結果

ラスベガスタイプのアルゴリズム

[余談] 毎回 $O(j-i)$ 時間かければちょうど中央の値を見つけることもできる。

– QuickSort は実用上、最速と言われることが多いが、、、
  - a[m] がいつでも a[i]..a[j] の中央の値だと
    $T(n) \leqq 2T(n/2)+(c+1)n$
    が成立するので、$T(n)=O(n \log n)$ を得る。
  - a[m] がいつでも a[i] や a[j] だと
    $T(n) \leqq T(1)+T(n-1)+(c+1)n$
    が成立するので、$T(n)=O(n^2)$ を得る。

平均的な場合はどうなのか？

確率解析の例(1): QuickSortの解析

– QuickSort は実用上、最速と言われることが多いが、、、
  - 平均的には、a[i] … a[j] の値が一様に選ばれると仮定する。
    – つまり k 番目のものを pivot にする確率は $1/(j-i+1)$

[定理] 上記の仮定のもとでの QuickSort の実行時間の期待値の上界は $2n\,H(n) \sim 2n \log n$

オーバーヘッドの少なさから、確かに速いと言えそう

    – 記法
      » a[1]…a[n] の中で k 番目に来るべき要素を s_k と書く。
      » 指示変数(indicator variable) X_{ij} を以下のように定義する

$$X_{ij}=\begin{cases}0 & s_i \text{ と } s_j \text{ がアルゴリズム中で比較されないとき}\\ 1 & s_i \text{ と } s_j \text{ がアルゴリズム中で比較されるとき}\end{cases}$$

    – QuickSort の実行時間~要素の比較回数= $\sum_{i=1}^{n}\sum_{j>i} X_{ij}$

確率解析の例(1): QuickSortの解析

[定理] 仮定のもとでの QuickSort の実行時間の期待値の上界は $2n\,H(n) \sim 2n \log n$

（期待値の線形性による）

- QuickSort の実行時間の期待値= $E[\sum_{i=1}^{n}\sum_{j>i} X_{ij}] = \sum_{i=1}^{n}\sum_{j>i} E[X_{ij}]$
- 「p_{ij} =『s_i と s_j が比較される確率』」と定義すると、
  $E[X_{ij}] = p_{ij} \times 1 + (1 - p_{ij}) \times 0 = p_{ij}$
  よって p_{ij} の値を考える
- s_i と s_j はどんなときに比較されるのか？
  1. どちらかが pivot に選ばれている
  2. それまでの計算過程で、別々の qsort にわけられていない
     ⇔ s_i と s_j の間の要素が、まだ pivot として選ばれていない

確率解析の例(1): QuickSortの解析

[定理] 仮定のもとでの QuickSort の実行時間の期待値の上界は $2n\,H(n) \sim 2n \log n$

- s_i と s_j はどんなときに比較されるのか？
  1. どちらかが pivot に選ばれている
  2. それまでの計算過程で、別々の qsort にわけられていない
     ⇔ s_i と s_j の間の要素が、まだ pivot として選ばれていない
  3. $s_i, s_{i+1}, s_{i+2}, \ldots, s_{j-1}, s_j$ が pivot に選ばれる順番は、すべて等確率！
  4. よってこれらの中で s_i か s_j が最初の pivot になる確率: $\frac{2}{j-i+1}$
  - よって QuickSort の実行時間の期待値=

$$E[\sum_{i=1}^{n}\sum_{j>i} X_{ij}] = \sum_{i=1}^{n}\sum_{j>i} E[X_{ij}] = \sum_{i=1}^{n}\sum_{j>i} p_{ij} = \sum_{i=1}^{n}\sum_{j>i} \frac{2}{j-i+1}$$
$$= \sum_{i=1}^{n}\sum_{k=2}^{n-i+1} \frac{2}{k} \le 2\sum_{i=1}^{n}\sum_{k=1}^{n} \frac{1}{k} = 2nH(n)$$

確率解析の基本ツール(1):マルコフの不等式

- 定理 (マルコフの不等式)
  非負の値をとる任意の確率変数を Y とする。
  すると任意の正の実数 t に対して次が成立する: $\Pr[Y \ge t] \le \frac{E[Y]}{t}$
  これは次と同値である: $\Pr[Y \ge kE[Y]] \le \frac{1}{k}$
  - マルコフの不等式の意味とは…
    「期待値の k 倍以上になる確率は $1/k$ 以下である」
    - QuickSort の実行時間が $4n \log n$ 以上になる確率は ½ 以下
    - 10種類のクーポンを全部集めるために 69 回以上買う確率は 1/3 以下
      …
  - Yが非負であることと期待値 $E[Y]$ しかわからないときは、これは改善できない。分散もわかると、もうちょっと改善できる。

確率解析の基本ツール(1):マルコフの不等式

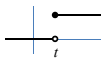
- 定理 (マルコフの不等式)
  非負の値をとる任意の確率変数を Y とする。
  すると任意の正の実数 t に対して次が成立する: $\Pr[Y \ge t] \le \frac{E[Y]}{t}$
  これは次と同値である: $\Pr[Y \ge kE[Y]] \le \frac{1}{k}$

[証明] 関数 $f(y)$ を次のように定義する: $f(y) = \begin{cases} 0 & y<t \\ 1 & y \ge t \end{cases}$

すると $\Pr[Y \ge t] = E[f(Y)]$ である。

ここですべての y に対して $f(y) \ge y/t$ なので、

$$\Pr[Y \ge t] = E[f(y)] \le E[Y/t] = \frac{E[Y]}{t}$$



となる。

確率解析の基本ツール(1):マルコフの不等式

- 定理 (マルコフの不等式)
  非負の値をとる任意の確率変数を Y とする。
  すると任意の正の実数 t に対して次が成立する: $\Pr[Y \ge t] \le \frac{E[Y]}{t}$
  これは次と同値である: $\Pr[Y \ge kE[Y]] \le \frac{1}{k}$

演習問題2:

正整数 k を一つ固定する。このとき非負の値をとる確率変数 Xで以下の条件を満たすのはどのような確率変数だろうか。

$$\Pr[X \ge kE[X]] = \frac{1}{k}$$

確率解析の基本ツール(2):チェビシェフの不等式

- 定理 (チェビシェフの不等式)
  期待値 μ_x と標準偏差 σ_x をもつ確率変数を X とする。
  すると任意の正の実数 t に対して次が成立する: $\Pr[|X - \mu_x| \ge t\sigma_x] \le \frac{1}{t^2}$
  - 復習:標準偏差とは…
    - 定義: 確率変数Xが値 $x_1, x_2, \ldots, x_n$ をそれぞれ確率 $p_1, p_2, \ldots, p_n$ でとるとする。このとき期待値（平均値）μ_xと分散σ_x^2は次で定義される:

$$\mu_x = \sum_{i=1}^{n} p_i x_i$$
$$\sigma_x^2 = \sum_{i=1}^{n} p_i (x_i - \mu_x)^2$$

    - 分散の正の平方根σ_xが標準偏差
  - 標準偏差（分散）がわかっている確率変数に対しては、こちらの方がマルコフの不等式よりもずっと強力である。

確率解析の基本ツール(2):チェビシェフの不等式

– 定理(チェビシェフの不等式)
期待値 μ_x と標準偏差 σ_x をもつ確率変数を X とする。
すると任意の正の実数 t に対して次が成立する: $\Pr[|X-\mu_x| \geq t\sigma_x] \leq \frac{1}{t^2}$

[証明] まず、明らかに次が成立する。

$$\Pr[|X-\mu_x| \geq t\sigma_x] = \Pr[(X-\mu_x)^2 \geq t^2\sigma_x^2]$$

確率変数 Y を $Y=(X-\mu_x)^2$ とおくと、Y の期待値は σ_x^2 である。
Y と t^2 に対してマルコフの不等式を適用すると、

$$\Pr[Y \geq t^2 E[Y]] \leq \frac{1}{t^2}$$

より以下を得る。

$$\Pr[|X-\mu_x| \geq t\sigma_x] = \Pr[(X-\mu_x)^2 \geq t^2\sigma_x^2] \leq \frac{1}{t^2}$$

OHPシートで作った川崎ローズ
(デザイン:川崎敏和、
折った人:上原)

2回目の休憩

曲線折りの例
(デザイン、折った人:Martin Demaine)

- レポートを書いて提出。時間は30分くらい?

あのお父さん
(デザイン:上原、折った人:上原)

上原のところに
ある折り紙作品

たぶん世界最小前川デビル
(デザイン:前川淳、折った人:上原)

ソートの下界の話

– 比較に基づく任意のソートアルゴリズムは
$\Omega(n \log n)$ 時間の計算時間が必要である
- 証明(概略)
  – k 回の比較で区別できる場合の数は高々 2^k 種類しかない
  – n 個の要素の異なる並べ方は $n!$ 通りある
  – したがって少なくとも

$$2^k \geq n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

が成立していなければならない。
両辺の対数をとれば以下を得る。

$$k \geq \log\sqrt{2\pi n}\left(\frac{n}{e}\right)^n = n\log n - O(n) + \frac{1}{2}\log n + O(1)$$

超高速ソートの話

– 以下の特殊なソートを考える:
- 入力: a[1]…a[n] で、それぞれの a[i] の値は 1〜10
- 以下のアルゴリズムAは $O(n)$ 時間で動作する(!?)

1. 配列 b[1]=b[2]=…=b[10]=0; // b[i] は a[j]=i を満たす要素の個数
2. for i=1,2,…,n do b[a[i]]++;
3. for j=1,2,…,10 do "j をb[j]個出力する".

このソートAは比較に基づいていない!!

Radix sort, bucket sort などと呼ばれるソートと同様のアイデア

データが「整数」など「特殊」な場合はこちらの方が速い!!

データに
暗黙の仮定
があれば
利用できる
かもしれない

典型的なNP完全問題KNAP…を高速に解く方法(?)

– KNAP:
Input: アイテムの配列 a[1], …, a[n], 大きさ k
Output: $\sum_{i \in S} a[i] = k$ を満たす集合 $S \subseteq \{1,\ldots,n\}$ が存在するか?

– アルゴリズムB:それぞれの $a[i]$ が正整数なら以下で解ける

1. b[1]=b[2]=…=b[k]=0;
2. for i=1,2,…,n do
   1. for j=k,k-1,…,2,1 do
      1. if b[j]>0 then b[j+a[i]]=1;
3. if b[t]>0 then "yes" else "no".

一般には k の値が n に対して
指数関数的に大きくなりうるので、
実は多項式時間アルゴリズムではない!!

b[]をリストにすれば、実数などでも
動作する。

– Bの実行時間は $O(nk)$ 時間

データが「整数」など「特殊」な場合や、
とりうる値の組合せの数(b[]の要素数)が
n の多項式で押さえられるときは速い!!

典型的なNP完全問題KNAP…を高速に解く方法(?)

– KNAP:
Input: アイテムの配列 a[1], …, a[n], 大きさ k
Output: $\sum_{i \in S} a[i] = k$ を満たす集合 $S \subseteq \{1,\ldots,n\}$ が存在するか?

– 演習問題:次のアルゴリズムB'は何を計算しているのか?

1. b[1]=b[2]=…=b[k]=0;
2. for i=1,2,…,n do
   1. for j=k,k-1,…,2,1 do
      1. if b[j]>0 then b[j+a[i]]=b[j+a[i]]+1;
3. if b[t]>0 then "yes" else "no".

近似アルゴリズム(Approximation Algorithm)

- 近似アルゴリズムの枠組:
  - 「Yes/No」タイプの決定問題を「最適化問題」に改造して考える。
    (**注意**)最適化問題は「最小化問題」と「最大化問題」がある
- 例:
  - VC(頂点被覆): 大きさ最小の頂点被覆を見つける
  - MaxSAT: 与えられた論理式のうち、なるべく多くの項を "True" にする
  - 巡回セールスマン問題: すべての都市をめぐる最小コストの経路を探す
    (グラフを「辺に重み(コスト)のついたグラフ」にして、全経路を通れることにする)
  - KNAP: 大きさ k 以下の組合せの中で最大のものを見つける
- 近似アルゴリズムの良さは「近似率」(と計算時間)で測る:
  - 最適な解を O* として、アルゴリズムの出力を O とすると、
    最小化問題の近似率=O/O*≧1
    最大化問題の近似率=O*/O≧1
  - 近似率はいつでも1以上で、近似率1のアルゴリズムは誤差のないアルゴリズム。

近似アルゴリズム(Approximation Algorithm)

- アルゴリズムの良さを「近似率」(と計算時間)で測る:
  - $\mathcal{NP}$困難問題(を最適化問題に翻訳したもの)は典型的には以下の3つのタイプに分類できる

1. 定数倍近似すら困難なもの(クラス$\mathcal{APX}$: この授業では扱わない)
   1. "$\mathcal{P}=\mathcal{NP}$が成立しない限り定数倍近似は存在しない問題"など
2. 適当な定数に対して定数倍近似が可能なもの(2倍近似アルゴリズムなど)
3. 任意の正定数 ε>0 に対して以下の条件を満たすアルゴリズムが存在する:
   1. n と 1/ε に対する多項式時間で動作する
   2. (1+ε)近似解を出す

   このアルゴリズム(群)を多項式時間近似スキーム
   (PTAS; Polynomial Time Approximation Scheme)と呼ぶ

近似アルゴリズム(Approximation Algorithm)

2. 2倍近似アルゴリズムの例
   - **頂点被覆問題**(VC)の最適化バージョン

     入力: 無向グラフ$G=(V,E)$

     出力: 最小の頂点被覆 S
   - アルゴリズムC:
     1. $S:=\Phi$;
     2. G の辺 $e=\{u,v\}$を適当に1本選ぶ
     3. u と v を S に入れる
     4. u,v につながっている辺を G からすべて削除する
     5. G に辺が残っていれば 2 に戻る

なぜか2倍を切れるかどうかが難しい問題が多い

SはGの「頂点被覆」: どの辺$\{u,v\}$に対しても$u\in S$または$v\in S$が成立

線形時間で動作するのは簡単なので省略

[定理] アルゴリズムCの実行時間はO($|V|+|E|$)時間である。
また G の最適な頂点被覆を S^* とし、
アルゴリズムCの出力を S とすると、以下が成立:
$|S|/|S^*|\leqq 2$

近似アルゴリズム(Approximation Algorithm)

2. 2倍近似アルゴリズムの例
   - アルゴリズムC:
     1. $S:=\Phi$;
     2. G の辺 $e=\{u,v\}$を適当に1本選ぶ
     3. u と v を S に入れる
     4. u,v につながっている辺を G からすべて削除する

SはGの「頂点被覆」: どの辺$\{u,v\}$に対しても$u\in S$または$v\in S$が成立

[定理] G の最適な頂点被覆を S^* とし、
アルゴリズムCの出力を S とすると、以下が成立:
$|S|/|S^*|\leqq 2$

[証明] ステップ2で選ばれた辺 e の集合を C とおく。

e はステップ4で削除されるため同じ辺が2度選ばれることはない。∴$2|C|=|S|$。

C は頂点を互いに共有しない辺の集合で、$e\in C$ のそれぞれについて
少なくとも一方の端点は S^* に入っていなければならない。∴$|C|\leqq|S^*|$

したがって $|S|/|S^*|\leqq 2|C|/|C|=2$ である。

近似アルゴリズム(Approximation Algorithm)

2. 2倍近似アルゴリズムの例
   - アルゴリズムC:
     1. $S:=\Phi$;
     2. G の辺 $e=\{u,v\}$を適当に1本選ぶ
     3. u と v を S に入れる
     4. u,v につながっている辺を G からすべて削除する

SはGの「頂点被覆」: どの辺$\{u,v\}$に対しても$u\in S$または$v\in S$が成立

[定理] G の最適な頂点被覆を S^* とし、
アルゴリズムCの出力を S とすると、以下が成立:
$|S|/|S^*|\leqq 2$

[演習問題] アルゴリズムCは2倍近似アルゴリズムであることが証明された。
C の近似率2はこれ以上改善できないことを示せ。
具体的に、無限に多くの n に対して、C の近似率がちょうど 2 であるような
n 頂点グラフの例を示せ。

近似アルゴリズム(Approximation Algorithm)

3. 多項式時間近似スキームの例
   - KNAPの最適化バージョン

     Input: アイテムの配列 a[1], ..., a[n], 大きさ k

     Output: $\sum_{i\in S} a[i]=k_S<k$ を満たす集合$S\subseteq\{1,\ldots,n\}$でもっとも近いもの
   - アルゴリズムD (アルゴリズムBも参照):
     1. L:=Φ; //実現できる和の近似値のリスト
     2. for i=1,2,...,
        1. L のそれぞれの要素 b に対して、b+a[i]≦k ならそれを L に登録
        2. L のいくつかの要素を「丸め」て、不要なら捨てる
     3. L の中の k 以下のもっとも大きな値 k' を出力する

[アイデア] 個々の値をそれに近い値に丸めて、値の種類を減らす

[ポイント] Dで以下の2点が満たされればよい:
1. Lのサイズがいつでも n の多項式
2. 出力 k' がよい近似解を与える

近似アルゴリズム(Approximation Algorithm)

3. 多項式時間近似スキームの例
   - KNAPの最適化バージョン

     Input: アイテムの配列 a[1], ..., a[n], 大きさ k

     Output: $\sum_{i\in S} a[i] = k_S < k$ を満たす集合$S\subseteq\{1,\ldots,n\}$で もっとも近いもの
   - [仮定]
     - L が小さい順で並んでいるとする
     - 正の定数εを固定する

     [丸めの詳細]
     - Lの中で $b_1<b_2\leqq(1+\varepsilon/2n)\,b_1$ なら b_2 を削除する

     [定理] アルゴリズムDはPTASである。
     つまり任意の正定数εに対して以下が成立する:
     1. n, $1/\varepsilon$ の多項式で動作する
     2. 近似率は $(1+\varepsilon)$

     [証明] 1,2 ともにちょっと計算が必要...

近似アルゴリズム(Approximation Algorithm)

- 補題13.1: 自然数列 $a_1=1, a_2, \ldots, a_n=k$ が $(a_{i+1})/a_i \geqq 1+t$ を満たすなら、次が成立: $n < \dfrac{1+t}{t}\ln k$

  [証明] $(1+t)^n < k$ より、$n < \log_{t+1} k$ である。公式 $\dfrac{x}{1+x} \leq \ln(1+x) \leq x$ を適用すると以下を得る。

  $$n < \log_{t+1} k = \frac{\ln k}{\ln(1+t)} \leq \frac{(1+t)}{t}\ln k$$

- 補題13.2: $0<\varepsilon<1$に対して $\left(1+\dfrac{\varepsilon}{2n}\right)^n \leq 1+\varepsilon$

  [証明] 以下の公式をつかう。

  [公式1] $\lim_{n\to\infty}\left(1+\dfrac{x}{n}\right)^n = e^x$ (この式は n に対して単調増加関数)

  [公式2] $|x|\leqq 1$ならば $1+x \leq e^x \leq 1+x+x^2$

  以上より $\left(1+\dfrac{\varepsilon}{2n}\right)^n \leq e^{\varepsilon/2} \leq 1+\dfrac{\varepsilon}{2}+\left(\dfrac{\varepsilon}{2}\right)^2 \leq 1+\varepsilon$

近似アルゴリズム(Approximation Algorithm)

- KNAPの最適化バージョンの多項式時間近似スキーム

  [定理13.2] アルゴリズムDはPTASである。
  つまり任意の正定数εに対して以下が成立する:
  1. n, $1/\varepsilon$ の多項式で動作する
  2. 近似率は $(1+\varepsilon)$

  [証明] 1) Lのサイズが n, $1/\varepsilon$ の多項式で抑えられればよい。
  ここで L の要素列 $b_1, b_2, \ldots$ は、$1\leqq b_1, b_i \leqq k, b_{i+1}/b_i > (1+\varepsilon/2n)$ を満たす。
  よって補題13.1より、
  Lのサイズ$< \log_{(1+\varepsilon/2n)} k = ((2n+\varepsilon)\log k)/\varepsilon < (2n\log k)/\varepsilon$
  となる。

近似アルゴリズム(Approximation Algorithm)

- KNAPの最適化バージョンの多項式時間近似スキーム

  [定理13.2] アルゴリズムDはPTASである。
  つまり任意の正定数εに対して以下が成立する:
  1. n, $1/\varepsilon$ の多項式で動作する
  2. 近似率は $(1+\varepsilon)$

  [証明] 2) アルゴリズムの出力 k' を構成する $a[]$ の要素集合が存在する。
  この集合を S' とする。
  つまり次が成立する: $\sum_{a[]\in S'} a[] = k'$
  ここで入力に対する
  最適な集合を S^* とし、$\sum_{a[]\in S^*} a[] = k^*$ とする。
  S^* のそれぞれの
  $a[]$ に対しては、それが L に存在しているか、それを代替したものがあるはず
  である。代替されている場合、最悪だと $a[]$ は以下の値 a' で代替されている。

  $$\frac{a[]}{(1+\varepsilon/2n)^n} < \frac{a[]}{(1+\varepsilon/2n)^{n-1}} \leq a' < a[]$$

  よって
  $k^*/(1+\varepsilon/2n)^n \leqq k'$ が成立し、補題13.2より $k^*/k' \leqq 1+\varepsilon$ を得る。