算法数理工学
第13回

定兼　邦彦

チューリングマシン
TURING MACHINES

- チューリングマシンは1つのデータ構造を持つ
  - シンボルの文字列(テープ)
- プログラムが可能な操作
  - テープ上のカーソルを1つ左か右に動かす
  - テープの現在のカーソルの位置にシンボルを書く
  - シンボルに従って左右に動く
- 非常に限定された動作しかできないが，任意のアルゴリズムやプログラミング言語を模倣できる

- 定義:チューリングマシンは4つ組 $M = (K, \Sigma, \delta, s)$
  - K は状態の有限集合
  - $s \in K$ は初期状態
  - Σ はシンボルの有限集合 (M のアルファベットと呼ぶ)
  - δ: $(K \times \Sigma) \to (K \cup \{\mathrm{h}, \mathrm{Y}, \mathrm{N}\}) \times \Sigma \times \{\leftarrow, \rightarrow, —\}$
- Σ は特殊シンボルを含む
  - ▢: 空白
  - ▷: 最初の文字 (テープの左端を表す)
- h: 停止状態
- Y: 受理状態
- N: 拒否状態
- $\leftarrow$, $\rightarrow$, —: カーソルを左/右に動かす/動かない

- 関数 δ はチューリングマシンのプログラムである
- 現在の状態 $q \in K$ とカーソル位置の文字 $\sigma \in \Sigma$ から，3つ組 $\delta(q, \sigma) = (p, \rho, D)$ を返す
  - p: 次の状態
  - ρ: σ に上書きされるシンボル
  - D: カーソルが動く方向
- $\delta(q, \triangleright) = (p, \rho, D)$ ならば $\rho = \triangleright$, $D = \rightarrow$ とする (テープの左端なら必ず右に行き，$\triangleright$ は消えない)
- 初めは状態は s で，テープは $\triangleright$ の後に有限長の文字列 $x \in (\Sigma - \{\square\})^*$ が続く
- x をチューリングマシンの入力と呼ぶ
- カーソルの初期位置は $\triangleright$

- チューリングマシンは関数 δ に従って状態を変え，シンボルをテープに書き，カーソルを移動することを繰り返す．
- カーソルはテープの左端より左には行かないが，入力文字列の右端よりも右に行くことがある．
  その場合はシンボル $\square$ が書かれているとみなす．
- 文字列が長くなることを許すことは汎用的な計算を行うために必要．短くなることは無い．
- 3つの停止状態 h, Y, N のどれかになった場合，チューリングマシンは停止したと言う．
  - Y: 入力を受理した
  - N: 入力を拒否した

- チューリングマシン M が入力 x を受理 (Y) または拒否 (N) したとき，$M(x) = \mathrm{Y}$ (N) と書く.
- 状態が h で停止した時，$M(x) = y$ と書く
  (y は現在の文字列)
- M はある入力 x に対しては停止しないことがある.
  その時は $M(x) = \nearrow$ と書く.

オートマトンの例1

$M = (K, \Sigma, \delta, s)$, $K = \{s, q, q_0, q_1\}$, $\Sigma = \{0, 1, \square, \triangleright\}$

p ∈ K	σ ∈ Σ	δ(p, σ)
s	0	$(s, 0, \rightarrow)$
s	1	$(s, 1, \rightarrow)$
s	$\square$	$(q, \square, \leftarrow)$
s	$\triangleright$	$(s, \triangleright, \rightarrow)$
q	0	$(q_0, \square, \rightarrow)$
q	1	$(q_1, \square, \rightarrow)$
q	$\square$	$(q_0, \square, —)$
q	$\triangleright$	$(h, \square, \rightarrow)$
q_0	0	$(s, 0, \leftarrow)$
q_0	1	$(s, 0, \leftarrow)$
q_0	$\square$	$(s, 0, \leftarrow)$
q_0	$\triangleright$	$(h, \triangleright, \rightarrow)$
q_1	0	$(s, 1, \leftarrow)$
q_1	1	$(s, 1, \leftarrow)$
q_1	$\square$	$(s, 1, \leftarrow)$
q_1	$\triangleright$	$(h, \triangleright, \rightarrow)$

0. s, ▷010
1. s, ▷010
2. s, ▷010
3. s, ▷010
4. s, ▷010 ▢
5. q, ▷010 ▢
6. q_0, ▷01 ▢ ▢
7. s, ▷01 ▢ 0
8. q, ▷01 ▢ 0
9. q_1, ▷0 ▢ ▢0
10. s, ▷0 ▢ 10
11. q, ▷0 ▢ 10
12. q_0, ▷▢ ▢ 10
13. s, ▷▢ 0 10
14. q, ▷▢ 0 10
15. h, ▷▢ 0 10

$M(010) = \square\ 010$

入力を右にずらす

オートマトンの例2

$$M = (K, \Sigma, \delta, s),\ K = \{s, q\},\ \Sigma = \{0, 1, \square, \triangleright\}$$

オートマトンのコンフィギュレーションは
(状態，カーソルから左の文字列，
カーソルより右の文字列)で表現できる

n の2進表現から n+1 の2進表現を
求めるプログラム

$p \in K$	$\sigma \in \Sigma$	$\delta(p, \sigma)$
s	0	$(s, 0, \rightarrow)$
s	1	$(s, 1, \rightarrow)$
s	$\square$	$(q, \square, \leftarrow)$
s	$\triangleright$	$(s, \triangleright, \rightarrow)$
q	0	(h, 1, —)
q	1	$(q, 0, \leftarrow)$
q	$\square$	
q	$\triangleright$	$(h, \square, \rightarrow)$

$(s, \triangleright, 11011) \rightarrow (s, \triangleright 1, 1011) \rightarrow (s, \triangleright 11, 011) \rightarrow (s, \triangleright 110, 11) \rightarrow$
$(s, \triangleright 1101, 1) \rightarrow (s, \triangleright 11011, \varepsilon) \rightarrow (s, \triangleright 11011\ \square, \varepsilon) \rightarrow (q, \triangleright 11011, \square) \rightarrow$
$(q, \triangleright 1101, 0\square) \rightarrow (q, \triangleright 110, 00\square) \rightarrow (h, \triangleright 111, 00\square)$ $M(11011) = 11100$

$(s, \triangleright, 11) \rightarrow (s, \triangleright 1, 1) \rightarrow (s, \triangleright 11, \varepsilon) \rightarrow (s, \triangleright 11\ \square, \varepsilon) \rightarrow (q, \triangleright 11, \square) \rightarrow$
$(q, \triangleright 1, 0\square) \rightarrow (q, \triangleright, 00\square) \rightarrow (h, \triangleright 0, 0\square)$ $M(11) = 00$ オーバーフロー

多テープチューリングマシン

- k-テープチューリングマシンは，k 個のテープを持つチューリングマシン
- テープごとに異なるカーソルを持つ
- $M = (K, \Sigma, \delta, s)$
- 遷移関数 δ は現在の状態と k 個のカーソルの位置にあるシンボルから決まり，k 個の異なるシンボルを書き込める
- 文字列を出力するチューリングマシンの場合，最後のテープの文字列を出力とする.

定理: $f(n)$ 時間で動作する任意のk-テープチューリングマシン M に対し, $O(k^2\{f(n)\}^2)$ 時間で動作する1-テープチューリングマシン M' で, 任意の入力 x に対し　$M(x) = M'(x)$ となるものが存在する.

この定理は1-テープチューリングマシンを計算モデルとして用いることの1つの根拠になっている.

M で多項式時間で計算できるならばそれは M' でも多項式時間で計算できる.

RANDOM ACCESS MACHINES

- ランダムアクセスマシン (RAM) は，データ構造の上で動作するプログラムである.
- データ構造はレジスタの配列 (メモリ)
  - それぞれは任意桁の整数を格納可
- RAMのプログラム $\Pi = (\pi_1, \pi_2, \ldots, \pi_m)$ は有限長の命令の列
- RAMでの計算の各ステップでは，命令 π_K が実行される. K はプログラムカウンタ
- RAMは1-テープチューリングマシンで模倣できる.

Instruction	Operand	Semantics
READ	j	R[0] := I[j]
READ	R[j]	R[0] := I[R[j]]
STORE	j	R[j] := R[0]
STORE	R[j]	R[R[j]] := R[0]
LOAD	x	R[0] := x
ADD	x	R[0] := R[0] + x
SUB	x	R[0] := R[0] – x
HALF		R[0] := $\lfloor$R[0]/2$\rfloor$
JUMP	j	K := j
JPOS	j	if R[0]>0 K := j
JZERO	j	if R[0]=0 K := j
JNEG	j	if R[0]<0 K := j
HALT		K := 0

x は j, R[j], R[R[j]] のいずれか
I は入力文字列 (書き換え不可)

定理:あるチューリングマシンがある関数を $f(n)$ 時間で計算するとする. その関数を $O(f(n))$ 時間で計算するRAMプログラムが存在する.

定理:あるRAMプログラムがある関数を$f(n)$ 時間で計算するとする. その関数を $O(\{f(n)\}^3)$ 時間で計算する7-テープチューリングマシンが存在する.

1. I[1] I[2] … 入力文字列 (書き変えない)
2. b(1): b(R[1])); b(2): b(R[2]));…レジスタの2進表現
3. K プログラムカウンタ
4. b(i) アドレスレジスタ (テープ2から R[i] を読むときに使う)
5. x 6. y 算術演算 z := x+y 等を行う際に使う
7. z 出力文字列 R[0] もここに格納する

非決定性チューリングマシン
Non-deterministic Turing Machines

- 非決定性チューリングマシンは4つ組
  $N = (K, \Sigma, \Delta, s)$
  - K, Σ, s はチューリングマシンと同じ
  - Δ は関数ではなく“関係”
  - $\Delta \subset (K \times \Sigma) \rightarrow (K \cup \{\mathrm{h}, \mathrm{Y}, \mathrm{N}\}) \times \Sigma \times \{\leftarrow, \rightarrow, \text{—}\}$
- つまり，ある状態とシンボルから遷移する先が複数あり，その中の1つが選ばれる
- 入力が受理されるとは，状態が Y になるある遷移の列が存在することである.
- そのような遷移列が一つもないときだけ拒否される

- 言語 L は文字列の集合 ($L \subset \Sigma^*$)
- チューリングマシンM がL を認識する(M decides L)
  とは，任意の $x \in L$ に対し M が x を受理し，
  任意の $y \notin L$ に対し M が y を拒否すること
- 決定性(非決定性)チューリングマシン M (N) が
  $f(|x|)$ 時間で言語 L を認識するとは，
  M (N) が L を認識し，任意の $x \in \Sigma^*$ に対して
  M (N) が $f(|x|)$ 時間で停止すること

- 定義: NTIME($f(n)$) は非決定性チューリングマシンで $f(n)$ 時間で認識される言語の集合
- 計算量クラス NP は全ての NTIME(n^k) の和集合

- 定義: TIME($f(n)$) は決定性チューリングマシンで $f(n)$ 時間で認識される言語の集合
- 計算量クラス P は全ての TIME(n^k) の和集合

- P vs. NP 問題とは，$P = NP$ か $P \neq NP$ かを決定する問題で，未解決

- 命題: $P \subseteq NP$
- 証明: 決定性チューリングマシンは非決定性チューリングマシンの部分クラス.
(決定性チューリングマシンの遷移関数 δ は
非決定性チューリングマシンの関係 Δ で表せる)
- 定理: 言語 L が非決定性チューリングマシン N で $f(n)$ 時間で認識できるとする. するとある決定性3-テープチューリングマシン M で $O(c^{f(n)})$ 時間で L を認識できる (c は N に依存する定数). つまり

$$\mathrm{NTIME}(f(n)) \subseteq \bigcup_{c>1} \mathrm{TIME}\left(c^{f(n)}\right)$$

- 略証: M は N の非決定的選択を短い順に列挙し, 各入力に対して N の動作をシミュレートする.

還元 (REDUCTION)

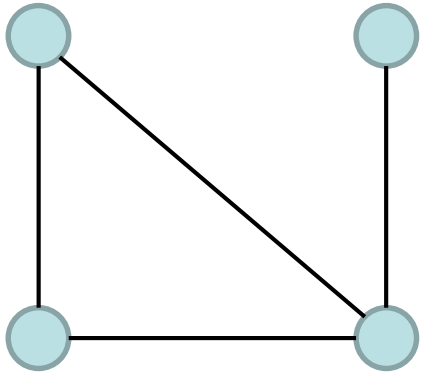
- 定義: 言語 L_1 が言語 L_2 に還元可能 (reducible) とは，以下の条件を満たす関数 R が存在すること
  - R は文字列から文字列への関数
  - 任意の入力 x に対し $x \in L_1 \Leftrightarrow R(x) \in L_2$
  - R はある決定性チューリングマシンで $O(\log |x|)$ 領域で計算可能
- R を L_1 から L_2 への還元または帰着と呼ぶ

- 命題: R をチューリングマシン M で計算される還元とすると，任意の入力に対し，M は多項式ステップで停止する.
- 証明: 入力 x に対する M のコンフィギュレーションは $\mathrm{O}(n\, c^{\log n})$ しかない ($n = |x|$).
  - x に対するヘッドの位置: n 通り
  - 計算中のテープの状態: $c^{\log n}$ 通り
- チューリングマシンは決定性なので，計算中に同じコンフィギュレーションが現れることは無い (現れたらマシンが停止しないことになる).
- つまり，計算ステップ数はコンフィギュレーションの数以下で，$\mathrm{O}(n^k)$ (k はある定数)

- 問題Bを問題Aに帰着する (B reduces to A) とは, Bの入力 (インスタンス) xのAの等価なインスタンスへの変換Rが存在すること.
- 入力xに対して問題Bを解くには, $R(x)$を計算し, それに対して問題Aを解けばいい.
- 問題Bを問題Aに帰着できることを$B \propto A$と書く.

ハミルトンパス問題

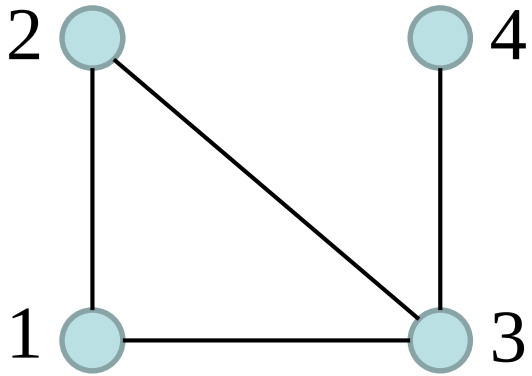
- 問題 (HAMILTON PATH): 入力グラフに,
  各節点をちょうど一回訪れるパスが存在するか.



巡回セールスマン問題

- 問題: (Traveling Salesperson Problem, TSP)
  n 点 1,2,…,n と2点間の非負整数の距離 d_{ij} が与えられたとき，全点を回る最短ツアーを求める問題.
- つまり，点の順列 π で $\sum_{i=1}^{n} d_{\pi(i),\pi(i+1)}$ が最小になるものを求める問題.
- 判定版 (decision version)
- 問題 (TSP(D))
  距離行列 d_{ij} の他に整数 B が与えられたときに，長さ B 以下のツアーがあるか決定する問題.

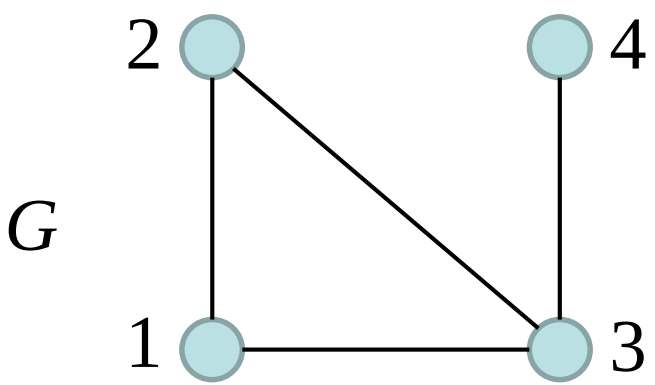
- 補題: ハミルトンパス問題は TSP(D) に帰着可能 (HAMILTON PATH $\propto$ TSP(D))
- 証明: ハミルトンパス問題の入力グラフ G に対し, TSP(D) の入力である距離行列 d_{ij} と整数B を決め G がハミルトンパスを持つとき，またその時に限り長さ B 以下のツアーがあるようにする.
- G の各点に対し，点を1つ作成する.
- G に枝 (i,j) があるとき $d_{ij} = 1$, ないとき $d_{ij} = 2$ とする
- $B = n+1$ とする.



G

$$d_{ij} \quad \begin{pmatrix} 2 & 1 & 1 & 2 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 1 \\ 2 & 2 & 1 & 2 \end{pmatrix} \qquad B = 5$$

- ツアーに含まれる枝数は n である (TSPの定義)
- 長さ $B = n+1$ 以下のツアーがあるなら，長さ 2 の枝は高々1本しか含まない
- ツアーの中には長さ 1 の枝が $n-1$ 本連続しているところが必ずある $\Rightarrow$ ハミルトンパスになっている
- 逆に，ハミルトンパスが存在するなら長さ B のツアーが作れる
- 問題の変換は $O(\log n)$領域(多項式時間)で行える



$$d_{ij} \quad \begin{pmatrix} 2 & 1 & 1 & 2 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 1 \\ 2 & 2 & 1 & 2 \end{pmatrix} \qquad B = 5$$

論理関数

- 定義: 論理関数は次のうちのいずれか
  - (a) 論理変数 x_i
  - (b) 論理関数 ϕ_1 の否定 $\neg\phi_1$
  - (c) 2つの論理関数 ϕ_1 , ϕ_2 の和 $(\phi_1 \vee \phi_1)$
  - (d) 2つの論理関数 ϕ_1 , ϕ_2 の積 $(\phi_1 \wedge \phi_1)$
- x_i や $\neg x_i$ をリテラル (literal) と呼ぶ
- 定義: 論理関数 ϕ が和積標準形 (conjunctive normal form, CNF) とは, $\varphi = \bigwedge_{i=1}^{n} C_i$ で各 C_i が1つ以上のリテラルの和となっていること.
- ϕ が積和標準形 (disjunctive normal form, DNF) とは, $\varphi = \bigvee_{i=1}^{n} D_i$ で各 D_i が1つ以上のリテラルの積

- CNFの例

$$\varphi = \big((x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_1) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)\big)$$

- 定理: 任意の論理関数に対し，それと等価な CNF と DNF がある
- 証明: 真理値表で真に対応する節を集めたものは DNF. 真理値表で偽に対応する節を集めた DNF の否定をドモルガンの法則で変換すると CNF.

充足判定性問題 (SAT)

- ある論理関数が充足可能 (satisfiable) とは,
  ある真理値割り当て $T = (a_1, a_2, \ldots, a_n)$ が存在し,
  論理関数が真になることである. そうでないとき
  充足不能 (unsatisfiable) という.

- 充足判定性問題 (SATISFIABILITY, SAT)
  CNF論理関数が与えられたとき, それが充足可能か判定する.

- 定理: ハミルトンパス問題はSATに帰着可能 (HAMILTON PATH $\propto$ SAT)
- 証明: グラフ G が n 節点 1,2,…, n を持つとする. n^2 個の変数 x_{ij}: $1 \le i,j \le n$ を持つCNF式 R(G)を作る.
- $x_{ij} = 1$ は, 節点 j がハミルトンパスの i 番目に現れることを意味する
- 各 j に対し節 $(x_{1j} \vee x_{2j} \vee \ldots \vee x_{nj})$ を作る. これは各節点 j が必ずハミルトンパスに含まれることを表す
- 節点 j が同時に i 番目と k 番目になることは無い. これを表す節として $(\neg x_{ij} \vee \neg\, x_{kj})$ を入れる.
- 各 i に対し節 $(x_{i1} \vee x_{i2} \vee \ldots \vee x_{in})$ を作る. これはハミルトンパスの i 番目の点があることを表す

- 2つの節点 j, k が同時に i 番目になることは無い. これを表す節として $(\neg x_{ij} \vee \neg\, x_{ik})$ を入れる.
- G の枝では無いペア (i, j) に対し，ハミルトンパスでは i の直後に j が来ることは無い．これを表す節として，$(\neg x_{k\,i} \vee \neg\, x_{k+1\,j})$ を入れる $(k=1,\ldots,n-1)$.
- $R(G)$ はこれらの節の積.
- G がハミルトンパスを持ち，またその時に限り $R(G)$ は充足可能である.
- R は $\mathrm{O}(\log n)$ 領域で計算できる.

完全性

- 帰着可能性は推移的であるため，問題を難しさに関して並べることができる.
- この半順序における極大元を考える.
- 定義: C を計算量クラスとし，L を C に属する1つの言語 (問題) とする. L が C-完全 (C-complete) とは任意の言語 $L' \in C$ が L に帰着可能であること.
- 定義: クラス C が帰着に関して閉じている (closed under reductions) とは，L が L' に帰着可能で $L' \in C$ ならば，$L \in C$ が成り立つこと
- 命題: P と NP は帰着に関して閉じている.
- あるNP-完全問題が P に属するなら，P = NP

- 定理 (Cookの定理) SATはNP-完全
- NPの定義より，NPに属する任意の問題はNTMで多項式時間で解ける．よって，任意のNTMがSATでシミュレートできることを示せばいい．
- 変数
  - $Q[i,k]$: 時刻 i に状態 q_k の時に真
  - $H[i,j]$: 時刻 i にカーソルの位置が j の時に真
  - $S[i,j,k]$: 時刻 i に j 番目のシンボルが s_k の時に真

制約

- G_1: 各時刻 i にちょうど1つの状態
  $\Rightarrow (Q[i,0] \vee \ldots \vee Q[i,r])$　少なくとも1つの状態
  $(\neg Q[i,k] \vee \neg Q[i,k'])\ (0 \le k < k' \le r)$ 高々1状態
- G_2: 各時刻にちょうど1つのカーソル位置
- G_3: 各時刻 i, 各位置 j にちょうど1つのシンボル
- G_4: 入力
  (Q[0, 0]): 時刻 0 に状態 0
  (H[0, 1]): 時刻 0 に位置 1
  S[0, 0, 0]: テープの左端は ▷
  $(S[0, h, k_h])\ (1 \le h \le |x| = n)$
  入力は $x = h_1\ h_2\ \ldots\ h_{|x|}$

- G_5: 停止状態
  $(Q[p(n), Y])$ 時刻 $p(n)$ で状態が Y
- G_6: 遷移関数 δ: $(K \times \Sigma) \rightarrow K \times \Sigma \times \{\leftarrow, \rightarrow, —\}$

  $(\neg H[i,j] \vee \neg Q[i,k] \vee \neg S[i,j,l] \vee \neg Q[i+1,k'])$
  時刻 i で状態が q_k, カーソル位置が j, j 番目のシンボルが s_l とすると, 時刻 $i+1$ では状態が $q_{k'}$

- NTMのプログラムが多項式時間 $p(n)$ で
  状態 Y で停止 $\Leftrightarrow$ 論理関数は充足可能

- 定理: ハミルトンパスはNP-完全
- 証明: SATをハミルトンパスに帰着 (略).

- 命題: TSP(D) $\in$ NP
- 証明: TSP(D) の入力を判定するNTMでは非決定的な選択として n 点の全てのツアーを考える.
  多項式時間で判定できる.

- 以上より
  SAT $\propto$ HAMILTON PATH $\propto$ TSP(D) $\propto$ SAT
  これらは全てNP-完全な問題

近似可能性 (Approximability)

- A を最適化問題とする. つまり, 各入力 x に対し, 実行可能解 (feasible solutions) の集合 $F(x)$があり, 各解 $s \in F(x)$ に対し正整数のコスト $c(s)$ が定まっている.
- 最適解は $\mathrm{OPT}(x) = \min_{s \in F(x)} c(s)$ と定義される. (最大化問題ならmax)
- アルゴリズム M が任意の入力 x に対して実行可能解 $M(x) \in F(x)$ を返すとする.
- M が ε-近似アルゴリズム ($\varepsilon \geq 0$) とは, 任意の x で

$$\frac{|c(M(x)) - \mathrm{OPT}(x)|}{\max\{\mathrm{OPT}(x), c(M(x))\}} \leq \varepsilon$$

最小化なら $c(M(x)) \leq \frac{1}{1-\varepsilon}\mathrm{OPT}(x)$

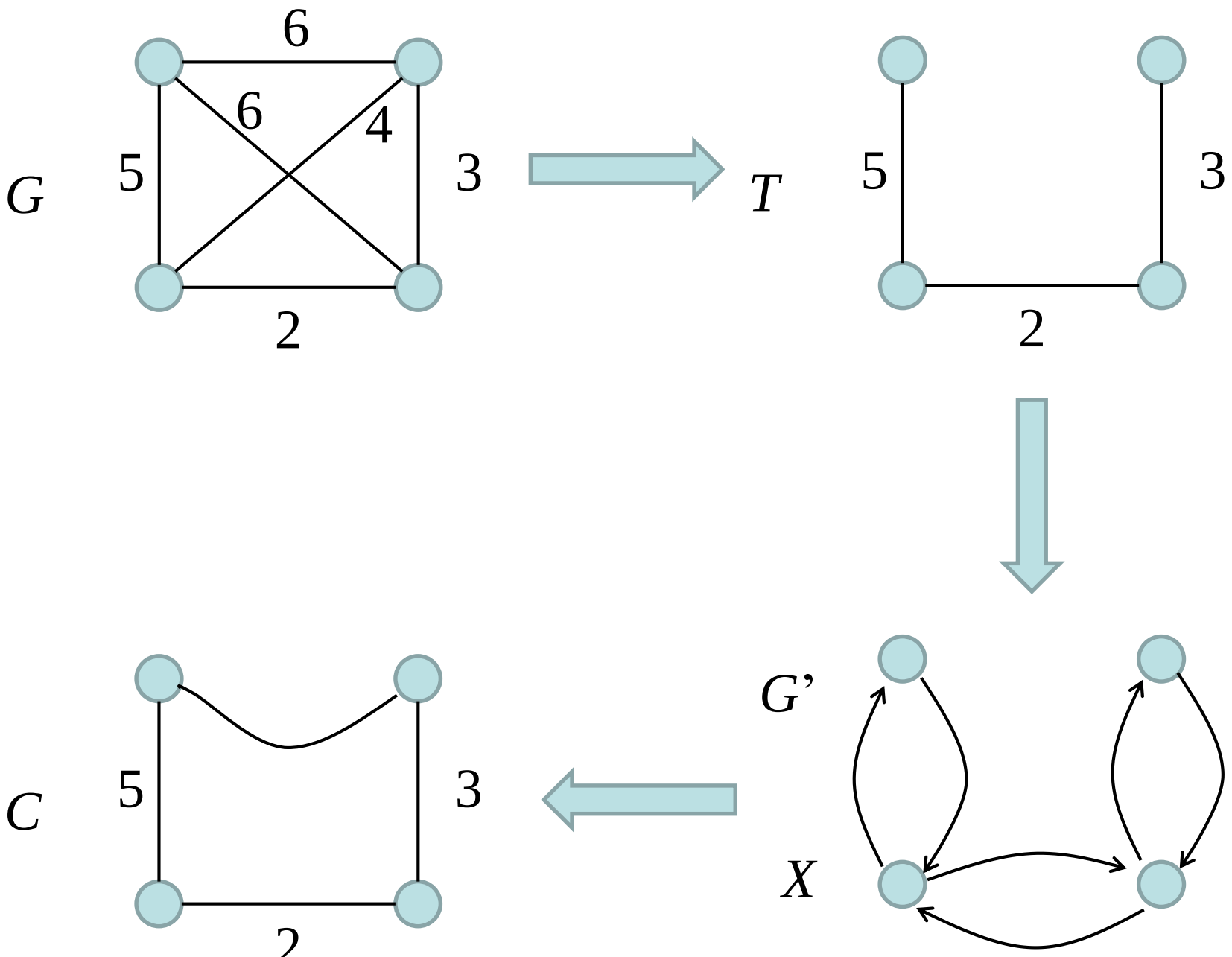
- $\varepsilon = 0$ ならば厳密アルゴリズム.
- ε が小さいと良い近似アルゴリズム.
- 定理: $\mathrm{P} \neq \mathrm{NP}$ と仮定する. 任意の $0 \leq \varepsilon < 1$ に対してTSPの多項式時間 ε-近似アルゴリズム は存在しない.
- 証明: ある $\varepsilon < 1$ に対して多項式時間 ε-近似アルゴリズムが存在したと仮定すると, NP-完全問題であるハミルトン閉路問題に対する多項式時間アルゴリズムが存在することになり矛盾.
- ハミルトン閉路問題のインスタンス $G = (V, E)$ に対し, $|V|$ 点のTSPのインスタンスを作る.

- 都市 i と j の距離は，G に枝 (i, j) があれば 1，なければ $|V| / (1-\varepsilon)$ とする．
- このTSPのインスタンスに対してε-近似アルゴリズムを実行する．コスト $|V|$ のツアーが見つかったら，それは G のハミルトン閉路になっている．
- 近似アルゴリズムの返した解が少なくとも1本の長さ $|V| / (1-\varepsilon)$ を含むなら，ツアーの総長は $|V| / (1-\varepsilon)$ より真に大きい．
- この解は ε-近似になっているはずなので，最適解のコストは近似解のコストの $1-\varepsilon$ 倍以上．つまり $\mathrm{OPT} > (1-\varepsilon) \cdot |V| / (1-\varepsilon) = |V|$ となり，G はハミルトン閉路を持たない．
- ハミルトン閉路が多項式時間で解けるので矛盾

- 定理: 枝のコストが三角不等式を満たすとき(メトリック),
  TSPに多項式時間 ½ -近似アルゴリズムが存在
- 三角不等式: 任意の i,j,k に対し $d_{ij} + d_{jk} \geq d_{ik}$

- 証明: 次の近似アルゴリズムを考える.

1. G の最小全域木 T を作る.
2. T の枝を全て2重にしたグラフ G' を作る.
3. G' のオイラー閉路 X を作る.
4. TSPの解として, X での順番で G の全点を訪れるツアー C を出力する.


G
6
6
4
5
3
2
T
5
3
2
G'
X
C
5
3
2

- $\mathrm{OPT} \geq c(T)$ である. なぜならば最適解から枝を1本取り除いたものは全域木で, そのコストは最小全域木のコスト以上だから.
- $c(X) = 2 \cdot c(T)$ (X は T の各枝を2回含む)
- $c(C) \leq c(X)$ (三角不等式より)
- 以上より $c(C) \leq 2\ \mathrm{OPT}$ つまり
  $\varepsilon = ½$ に対して $c(C) \leq \frac{1}{1-\varepsilon}\mathrm{OPT}$

- 定理: メトリックTSPに対する多項式時間 1/3-近似アルゴリズムが存在する.

$$c(C) \leq \frac{3}{2}\mathrm{OPT}$$

決定不能性 (Undecidability)

- 定義: 万能チューリングマシン (universal TM) U はチューリングマシンであり，入力として他のチューリングマシン M を符号化した文字列と，M への入力 x を繋げた文字列を取り，$U(M; x) = M(x)$ を計算するものである.
- つまり，U は M の x に対する動作を模倣する.
- 停止性問題 (The HALTING Problem): チューリングマシン M の記述とその入力 x が与えられたとき，M が x に対して停止するか否かを判定する.

- 定義: 言語 H を $H = \{M; x \mid M(x) \neq \nearrow\}$ と定義する
- つまり, H は全てのチューリングマシンと停止する入力を符号化した文字列の集合
- 定理: H を認識する万能チューリングマシンは存在しない.
- 証明: 背理法で示す. H を認識するチューリングマシン M_H が存在すると仮定する. M_H を変更し, 次の動作をするチューリングマシン D を作る.
  - 入力 M に対し, D は M_H が入力 $M;M$ に対して動作するのと同じように動作する. M_H が停止するまで繰り返す(仮定より M_H は必ず停止する).

  - M_H が入力を受理したとき，D はカーソルを右に動かし続ける状態に入る.
  - M_H が入力を拒否したとき，D は停止する.
- つまり

  $D(M) = \text{if } M_H(M;M) = \text{Y then} \nearrow \text{ else Y}$
- $D(D)$ はどうなるか？
  - $D(D) = \nearrow$ ならば，D の定義から $M_H(D;D) = \text{Y}$ つまり $D;D \in H$ となり，H の定義より $D(D) \neq \nearrow$
  - $D(D) \neq \nearrow$ ならば，$M_H(D;D) = \text{N}$ となり $D;D \notin H$. H の定義より $D(D) = \nearrow$
- どちらの場合も矛盾. よって H を認識するチューリングマシンは存在しない.
- このような証明は対角線論法と呼ばれる.