Chapter 9 (sec 9.2 / 9.1.1)

NP-complete

\[ \Rightarrow \text{hard problem} \]

Optimal?

Approximation Algorithms

That gives a guarantee of the "closeness" to optimal solution

Set cover

\[ \Rightarrow \text{greedy approach} \]

\[ \approx o(\log n) \]

\[ \text{approximation factor} \]

Optimization task

\[ \min \]

\[ \min \text{ TSP} \]

\[ \min \text{ vertex cover} \]
Given an instance $I$, let $\text{opt}(I)$ be the optimal value.

Given an also $A$, let $A(I)$ be the solution returned by $A$.

**Approx ratio**

$$\alpha(A) = \frac{A(I)}{\text{opt}(I)}$$

Over all possible instances:

$$\alpha(A) = \max_I \left\{ \frac{A(I)}{\text{opt}(I)} \right\}$$

**Maximiztion problem:**

- Max clique
- Max independent set

$$\alpha(A) = \frac{\text{opt}(I)}{A(I)} \geq 1$$

Value returned by $A$.

TSP: Travelling Salesman Problem
Given a graph \( G \), \( \{ d_{ij} \}_{i,j=1}^{n} \) is a set of all pairwise distances.

Minimize the cost of a cycle that starts and ends at an origin vertex and visits every other vertex exactly once.

1) \( d\text{-TSP} \in \text{NP-complete} \)

   decision version

2) \( \text{opt}\text{-TSP} \) is extremely hard

   \( \rightarrow \) "harder" than other \( \text{NP-hard} \) problems

\[ \alpha(A) = O\left(\log n\right) \]

Greedy (poly-time)

\( \text{TSP} \in \text{NP-hard} \)

No poly-time approx except... unless \( P = \text{NP} \)
d - TSP ∈ NP- Complete

known: (Rudrata cycle / hamilton cycle ∈ NPC)

Rudrata → TSP

1) TSP ∈ NP

given G, d_{ij}, b

bound

given a solution S

O(|S|) to choose a tour cost ≤ b

2) Rudrata → TSP

Rudrata: Given graph G, does there exist a path that visits all vertices (once), except for origin.
Rudrata

1) If Rudrata path exists in $G$, then $\exists$ a TSP with cost $n$ in $\text{TSP}(G)$

$$n = |V|$$

2) If Rudrata does not exist

\[ \text{we must choose an} \]
4) If a Hamiltonian cycle does not exist

\[ \Rightarrow \text{ we must choose an edge in } TSP(\delta) \text{ that is not in the original graph} \]

\[ \sum (1 + c) + (n-1) \]

\[ \geq n + c \]

\[ \Rightarrow \text{ there is no } TSP \text{ solution of value } \leq n \]

1) If a Hamiltonian cycle then \( TSP \leq n \)

2) If \( TSP \leq n \) then a Hamiltonian cycle

\[ \Rightarrow \]

Contrapositive

If no Hamiltonian cycle then \( TSP \geq n \)

Show that there is no poly-time approx. for \( TSP \)

Let \( A \) be a poly-time algo. for \( TSP \)

\[ A(I) \leftarrow \text{ value of } TSP \text{ using } A \]

\[ \alpha = \alpha(A) = \frac{A(I)}{\text{Opt}(I)} \]
\((\alpha)\) is approx ratio for \(A\).

Use this also \(A\) to solve the \(d\)-rudrata problem in poly-time.

Given \(G\), an instance for rudrata

choose \(C = n\alpha\)

\[\text{Is there a rudrata cycle?}\]

Given this \(Tsp(G)\) instance, \(wn A\) on this graph.

let \(A\) return a tour

\[\exists \text{ opt}(Tsp(G)) = (n)\]
\( \alpha = \frac{\text{opt}(I)}{\text{opt}(J)} \)

\( \alpha = \frac{\lambda(J)}{\lambda} \quad \Rightarrow \quad \text{opt}(J) = \alpha \cdot \lambda \)

b) If \( \text{opt}(G) \geq n + c \), then the optimal solution is \( n + c \).

Then \( A \) would return a solution

\( (n + c) \cdot \alpha \)

\( = n \alpha + c \cdot \alpha \)

\( = n \alpha + n \alpha \cdot \alpha \)

\( = n \alpha (1 + \alpha) \quad \alpha \geq 1 \)

\( > n \alpha \).

Then vadrot exists.

Then vadrot does not exist.
There is poly-time approx to TSP if $P = NP$.

If we can approximate TSP in poly-time, then

$\text{Truck cycle problem} \in P \implies P = NP$.

TSP cannot be approximated in polynomial-time

- general TSP
- \( d_{ij} > 0, \ \text{can be any value} \)

Special case: metric TSP

- \( d_{ij} \) satisfy certain properties
  1) \( d(x, y) > 0 \) and \( d(x, x) = 0 \)
  2) \( d(x, y) = d(y, x) \)
  3) triangle inequality

\[ d(x, z) \leq d(x, y) + d(y, z) \]
For metric $TSP$ there is a $\alpha = 2$ approx algorithm

\[
\frac{A(I)}{\text{opt}(I)} \geq \frac{A(I)}{\text{lower bound}} \leq 2
\]

$TSP$ graph

\[c - d - b - c\]

path through all vertices
any path is also a tree

\[ \text{Cost}(\text{path}) \geq \text{mst}(G) \]

given in poly-time

\[
\frac{A(I)}{\text{opt}(I)} \geq \frac{A(I)}{\text{mst}(G)} \leq 2
\]

Blue = mst for G

Cost of "double" tour

\[ = 2 \cdot \text{mst}(G) \]
Part III

Chapter 6:

Dynamic programming

how to define the recursive formula

\[ \mu(i) = \max_{j < i} \min \{ \mu(j) \} \]

\[ \mu(1) = \text{defined} \]

problem is term of subproblem,
Chapter 2

Network Flows

- max flow = min cut

- residual graph

- stopping criteria = what does it imply

Bipartite matching

Chapter 8

Reductions

Showing that a "new" problem is NP-complete

- TSP
- VC
- MIS
- CLIQUE
Chapter 9

approx ratio

\( \Gamma_{SP} \)

\( \Rightarrow \) why not be care in coding.

\( \geq 50\% \) of the questions on page 111

Part 1

\( \big-O \) (Chapter 0)

Chapter 1

\( \Rightarrow \) Euclid's also for gcd

\( \Rightarrow \) primality testing

\( \Rightarrow \) exponentiation trick \( \Rightarrow \) doubling

\( \Rightarrow \) divide-and-conquer

(no crypto stuff)

\( \Rightarrow \) hashing

Chapter 2

merge sort

\( \text{Vandemon relegated} \) \( \leftarrow \) quick sort
Randomized Selection ← quicksort
(no $T^2$ or polynomials)

Chap 3
DFS \rightarrow \text{top. order}

Chap 4
DFS \rightarrow \text{shortest paths}
Dijkstra's + PQ

Chap 5
Greedy methods
MST \rightarrow \text{union-find} \quad (\text{CC})
Huffman
Set cover ← greedy approx alg