

**ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ**



ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

M.Sc. Program in Computer Science Department of Informatics

**Design and Analysis of Algorithms
The Traveling Salesman Problem (TSP)**

**Vangelis Markakis
markakis@gmail.com**

Traveling Salesman Problem (TSP)

TSP

I: A complete directed weighted graph $G=(V,E)$, integer B

Q (Decision): Is there a Hamiltonian Cycle in G (a tour) of cost $\leq B$?

Optimization: Find a tour of minimum cost

One of the most well studied problems in Computer Science,
Operations Research, ...

Brute force approach: $O(n!)$ – No way!

NP-hardness

- We have seen already that TSP is NP-complete

Some interesting special cases:

- **Δ -TSP**: A special case of TSP where the triangle inequality holds, i.e., $w(i,k) \leq w(i,j) + w(j,k)$ $1 \leq i, j, k \leq n$
- **TSP(1,2)**: all weights equal to 1 or 2
- And many others...

Most interesting cases turn out to be NP-complete as well

Coping with NP-complete problems

1. Small instances
2. Special cases
3. Exponential algorithms (**Dynamic Programming, Branch and Bound,...**)
4. Approximation algorithms
5. Randomized algorithms
6. Heuristic algorithms

DP for TSP

We need to identify first the subproblems we will solve

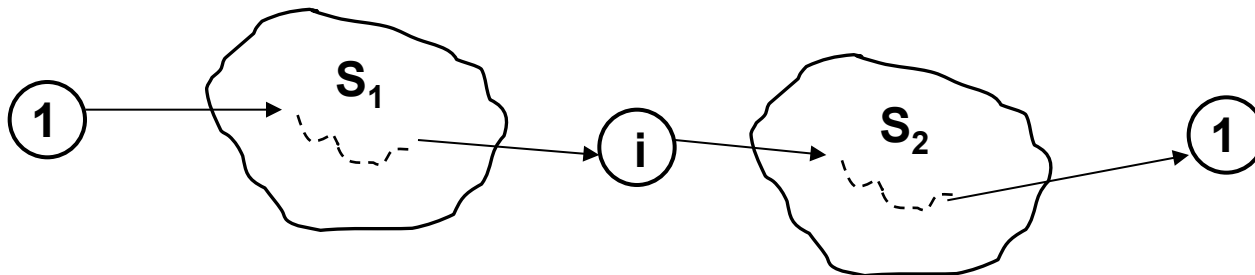
- Recall first the Hamiltonian path problem (HP)
- Variation in weighted graphs: find a HP of minimum weight

Optimal Substructure Property:

Assume w.l.o.g. that we start the tour at node 1

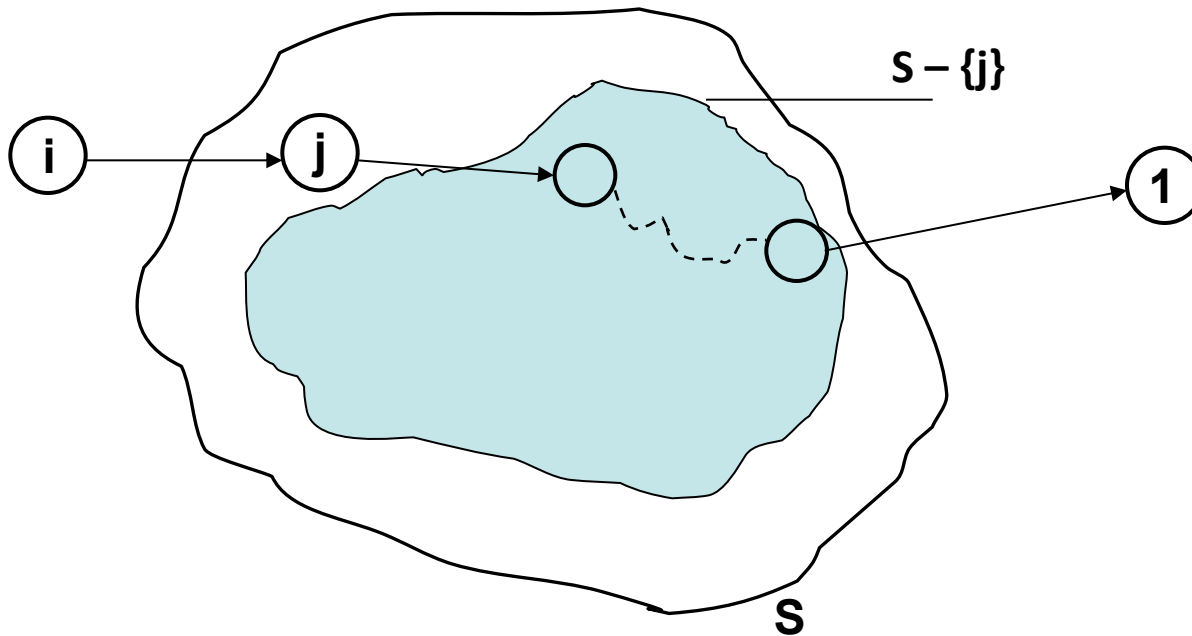
Assume that $1 \rightarrow \dots S_1 \dots \rightarrow i \rightarrow \dots S_2 \dots \rightarrow 1$ is an optimal tour

Then the path $i \rightarrow \dots S_2 \dots \rightarrow 1$ must be an optimal Hamiltonian path in $V - S_1$



DP for TSP

Let $g(i, S)$ = the cost of the shortest path $i \rightarrow \dots \rightarrow 1$, going from node i to node 1, using **all** the nodes of S (i.e., the minimum weight HP starting from i , in the graph induced by $S \cup \{i, 1\}$, $S \subset V$)

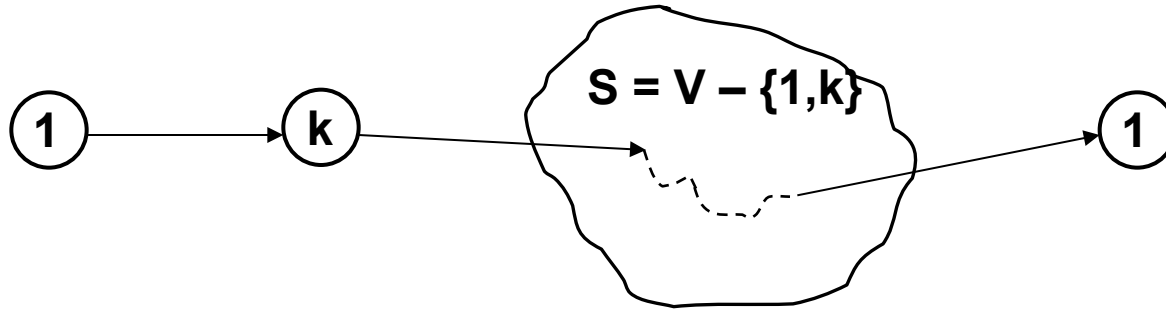


$$g(i, S) = \min_{j \in S} \{ w(i, j) + g(j, S - \{j\}) \}$$

DP for TSP

Our aim is to find

$$g(1, V - \{1\}) = \min_{2 \leq k \leq n} \{w(1, k) + g(k, V - \{1, k\})\}$$



How ?

By finding $g(k, V - \{1, k\})$ for all choices of k

This can be done by using the optimal substructure for $g(i, S)$

$$g(i, S) = \min_{j \in S} \{w(i, j) + g(j, S - \{j\})\}$$

DP for TSP

Obviously, $g(i, \emptyset) = w(i, 1)$

We can find $g(i, S)$ for all sets S , with $|S| = 1$

Then find $g(i, S)$ for all sets S , with $|S| = 2$

...

and then find $g(i, S)$ for all sets S , with $|S| = n-2$

Finally: $g(1, V - \{1\})$ --- $|S| = n-1$

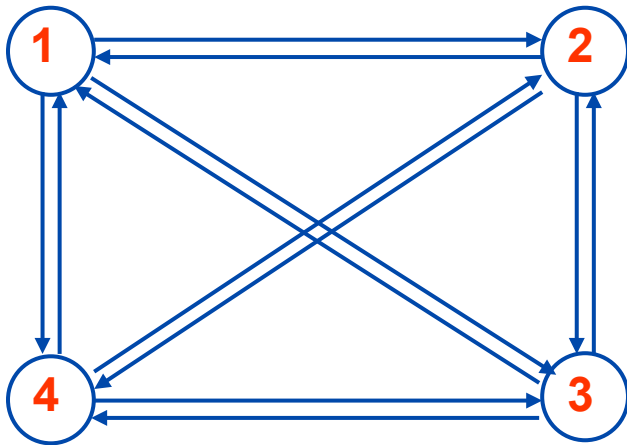
We need to compute $g(i, S)$

for **EVERY** set S of **EACH** possible size $|S| = 1, 2, \dots, n-2$,

and for all $i \in V - (S \cup \{1\})$

DP for TSP

Example



W :

| | | | |
|---|----|----|----|
| 0 | 10 | 15 | 20 |
| 5 | 0 | 9 | 10 |
| 6 | 13 | 0 | 12 |
| 8 | 8 | 9 | 0 |

DP for TSP

$|S|=0:$ $g(2,\emptyset)=5,$ $g(3,\emptyset)=6,$ $g(4,\emptyset)=8$

| | | |
|----------|---|---------------|
| $ S =1:$ | $g(2,\{3\}) = w_{23} + g(3,\emptyset) = 9 + 6 = 15$ | } $S = \{3\}$ |
| | $g(4,\{3\}) = 15$ | |
| | | |
| | $g(2,\{4\}) = 18$ | } $S = \{4\}$ |
| | $g(3,\{4\}) = 20$ | |
| | | |
| | $g(3,\{2\}) = 18$ | } $S = \{2\}$ |
| | $g(4,\{2\}) = 13$ | |

| | | |
|----------|--|-------------|
| $ S =2:$ | $g(2,\{3,4\}) = \min\{ w_{23} + g(3,\{4\}), w_{24} + g(4,\{3\}) \} = 25$ | $S=\{3,4\}$ |
| | $g(3,\{2,4\}) = \min\{ w_{32} + g(2,\{4\}), w_{34} + g(4,\{2\}) \} = 25$ | $S=\{2,4\}$ |
| | $g(4,\{2,3\}) = \min\{ w_{42} + g(2,\{3\}), w_{43} + g(3,\{2\}) \} = 23$ | $S=\{2,3\}$ |

| | | |
|---------------------------|--|---------------|
| $g(1,\{2,3,4\}) = \min\{$ | $w_{12} + g(2,\{3,4\}),$ $w_{13} + g(3,\{2,4\}),$ $w_{14} + g(4,\{2,3\}) \} =$ | $S=\{2,3,4\}$ |
| | $= \min\{35, 40, 43\} = 35$ | |

DP for TSP

```
for i = 2 to n do g(i,  $\emptyset$ ) = w(i, 1) ;
```

```
for k = 1 to n-2 do // for all sizes of S
```

```
  for each  $S \subseteq V - \{1\}$  s.t.  $|S|=k$  do // for all possible sets of size k
```

```
    for each  $i \in V - (S \cup \{1\})$ 
```

```
       $g(i, S) := \min_{j \in S} \{ w(i, j) + g(j, S - \{j\}) \} ;$ 
```

```
find g(1,  $V - \{1\}$ ) ;
```

DP for TSP

Complexity:

$N = \#$ of $g(i,S)$ computations

For each value of $|S|$ there are $\leq n - 1$ choices for i

The number of sets S with $|S| = k$ not including 1 and i is $\binom{n-2}{k}$

$$N = \sum_{k=0}^{n-2} (n-1) \binom{n-2}{k} = (n-1)2^{n-2}$$

$T(n) = N \cdot \text{time to compute } g(i,S) = N \cdot O(n)$

$T(n) = O(n^2 2^n)$, appropriate only for small instances

Coping with NP-complete problems

1. Small instances
2. Special cases
3. Exponential algorithms (Dynamic Programming, Branch and Bound,...)
4. Approximation algorithms
5. Randomized algorithms
6. Heuristic algorithms

Branch-and-Bound

A lower bound on the optimal solution: $\frac{1}{2} \sum_{i=1}^n (\min_{j \neq i} \{w_{i,j}\} + \min_{j \neq i} \{w_{j,i}\})$

- the half of the sum of minimum elements of each row and each column
- For every node one edge of the tour has to come towards i and one has to leave from i

Σ_0

| | A | B | C | D | |
|---|---|---|---|---|----------------------|
| A | x | 3 | 2 | 7 | 2 |
| B | 4 | x | 3 | 6 | 3 |
| C | 1 | 1 | x | 3 | 1 |
| D | 1 | 6 | 6 | x | 1 |
| | 1 | 1 | 2 | 3 | LB = 14/2 = 7 |

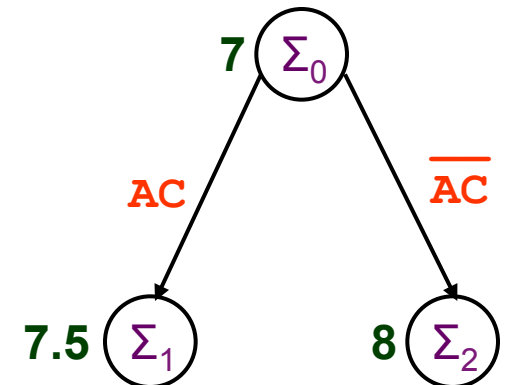
Branch-and-Bound

Σ_1 AC in tour \rightarrow CA, AB, AD, BC, DC not in tour (why ?)

| | A | B | C | D | |
|---|---|---|---|---|-------------------|
| A | x | x | 2 | x | 2 |
| B | 4 | x | x | 6 | 4 |
| C | x | 1 | x | 3 | 1 |
| D | 1 | 6 | x | x | 1 |
| | 1 | 1 | 2 | 3 | $LB = 15/2 = 7.5$ |

Σ_2 AC not in tour

| | A | B | C | D | |
|---|---|---|---|---|-----------------|
| A | x | 3 | x | 7 | 3 |
| B | 4 | x | 3 | 6 | 3 |
| C | 1 | 1 | x | 3 | 1 |
| D | 1 | 6 | 6 | x | 1 |
| | 1 | 1 | 3 | 3 | $LB = 16/2 = 8$ |



Branch-and-Bound

Σ_3

AC in tour \rightarrow CA, AB, AD, BC, DC not in tour

CB in tour \rightarrow CD, DB, BA not in tour

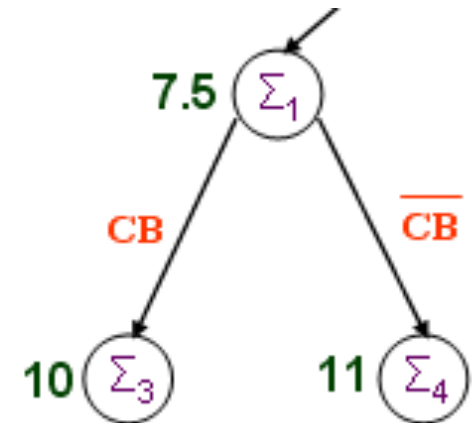
| | A | B | C | D | |
|---|---|---|---|---|-----------------------|
| A | x | x | 2 | x | 2 |
| B | x | x | x | 6 | 6 |
| C | x | 1 | x | x | 1 |
| D | 1 | x | x | x | 1 |
| | 1 | 1 | 2 | 6 | LB = 20/2 = 10 |

Σ_4

AC in tour \rightarrow CA, AB, AD, BC, DC not in tour

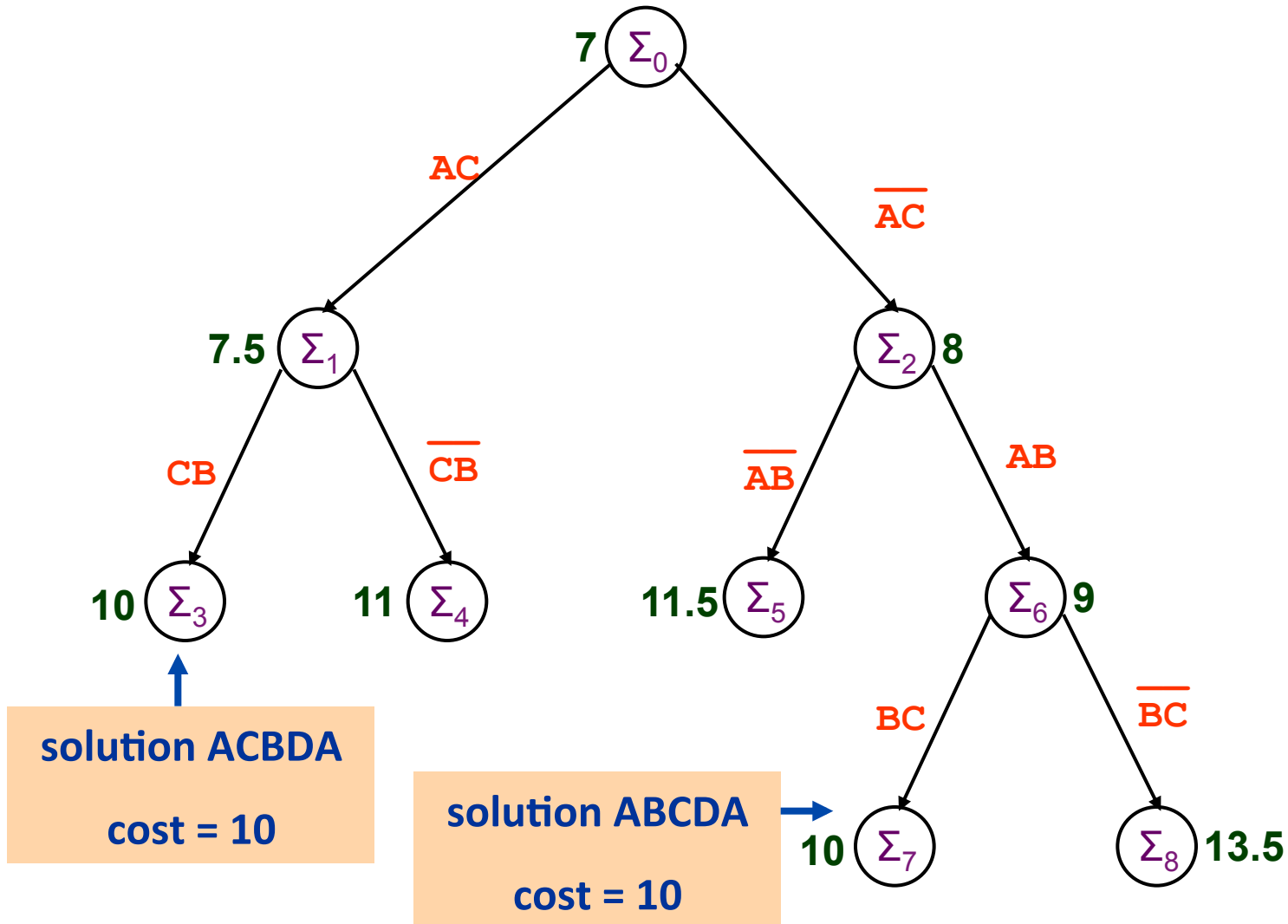
CB not in tour

| | A | B | C | D | |
|---|---|---|---|---|-----------------------|
| A | x | x | 2 | x | 2 |
| B | 4 | x | x | 6 | 4 |
| C | x | x | x | 3 | 3 |
| D | 1 | 6 | x | x | 1 |
| | 1 | 6 | 2 | 3 | LB = 22/2 = 11 |



and so on ...

Branch-and-Bound



Branch-and-Bound

Parameters

- Maintain a set S of active states
- Initially $S = \{\Sigma_0\}$ (nothing has been expanded yet)
- In each step extract state Σ from S (Σ is the state to be expanded)
- UB is a global upper bound of the optimum solution
 - For minimization problems we initially set $UB = +\infty$
- $LB(\Sigma)$ is a lower bound on all solutions represented by state Σ (i.e. From all solutions that can arise after expanding Σ)
- Whenever we reach a terminal node with $LB(\Sigma) \leq UB$, then we can update our current UB
- During the process, we do not need to examine further nodes where their LB is higher than UB!

Branch-and-Bound

Algorithm Branch and Bound

```
{ S = { $\Sigma_0$ };
  UB =  $+\infty$ 
  while S  $\neq$   $\emptyset$  do
  {   get a node  $\Sigma$  from S;
      //which node ? FIFO/LIFO/Best LB
      S := S - { $\Sigma$ };
      for all possible "1-step" extensions  $\Sigma_j$  of  $\Sigma$  do
      {   create  $\Sigma_j$  and find LB( $\Sigma_j$ );
          if LB( $\Sigma_j$ )  $\leq$  UB then
              if  $\Sigma_j$  is terminal then
                  {   UB := LB( $\Sigma_j$ );
                      optimum :=  $\Sigma_j$    }
                  else add  $\Sigma_j$  to S      }      }      }
```

Coping with NP-complete problems

1. Small instances
2. **Special cases**
3. Exponential algorithms
4. **Approximation algorithms**
5. Randomized algorithms
6. Heuristic algorithms

Approximability of TSP

Is there any $f(n)$ -approximation algorithm for TSP ? **NO !**

Theorem: For any (polynomial time computable) function $f(n)$ (with $f(n) \geq 1$ for all n), TSP cannot be approximated within a factor of $f(n)$, unless $P=NP$.

Proof:

Reduction from Hamilton Cycle (HC)

Claim: If there is an $f(n)$ -approximation algorithm A for TSP, then, there is a poly-time algorithm for HC

Consider an instance of HC, i.e., a graph $G=(V,E)$, with $|V| = n$

Construct a weighted graph $G' = (V, E')$, $E' = V \times V$ (all possible edges) with weights

$$w(u,v) = \begin{cases} 1, & \text{if } (u,v) \in E \\ n f(n), & \text{otherwise} \end{cases}$$

Approximability of TSP

Proof (cont.):

Running A on G' returns a tour of cost C

- a) if G is Hamiltonian,
 - optimal tour has $C^* = n$,
 - Algorithm A will return a tour with cost $C \leq nf(n)$

- b) if G is not Hamiltonian
 - The optimal tour must contain at least one edge of cost $n f(n)$
 - $C^* \geq nf(n) + (n-1) > n f(n)$
 - $C > nf(n)$

Δ -TSP: A 2-approximation

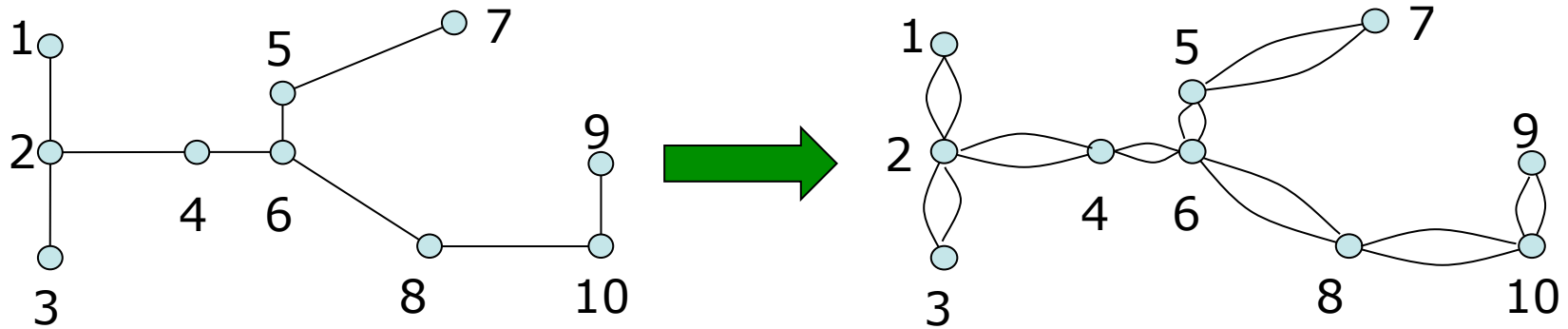
- Recall: Δ -TSP = special case of TSP where the triangle inequality holds, i.e., $w(i,k) \leq w(i,j) + w(j,k)$, $1 \leq i, j, k \leq n$
- A very natural special case, satisfied by many distance functions

Theorem: There exists a 2-approximation algorithm for Δ -TSP

- How do we start with designing an approximation algorithm?
- First and most important step: we need a lower bound on the cost of the optimal solution
- Consider an instance I of TSP
- **Claim:** $\text{OPT}(I) \geq \text{MST}(I)$
- Proof: delete one edge e from an optimal solution, what remains is a spanning tree F

$$\text{OPT}(I) = w(e) + C(F) \geq w(e) + \text{MST}(I) \geq \text{MST}(I)$$

Δ -TSP: A 2-approximation



Step 1: Find a minimum spanning tree, T , of G , of cost $C(T)$

Step 2: Double the edges of T and let T' be the obtained (multi)graph

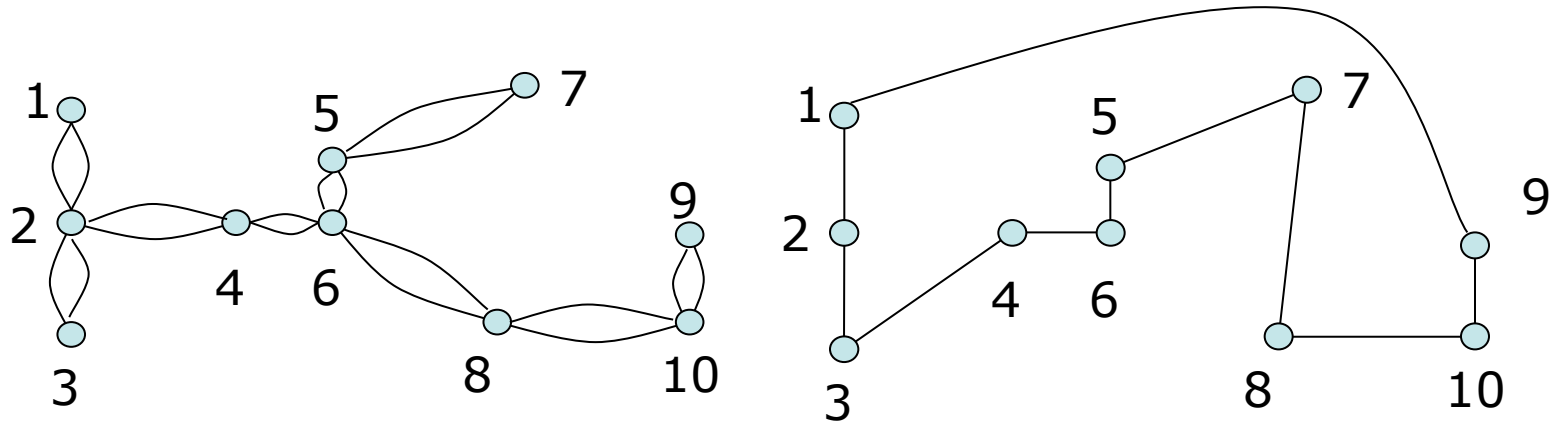
All vertices of T' are of even degree

Recall from graph theory:

- **Euler cycle:** A tour that visits all the edges exactly once
- A graph is Eulerian (i.e., has an Euler cycle) iff every vertex has an even degree

In the example: Euler cycle W : 1, 2, 3, 2, 4, 6, 5, 7, 5, 6, 8, 10, 9, 10, 8, 6, 4, 2, 1

Δ -TSP: A 2-approximation



Step 3: Find an Euler cycle W in T'

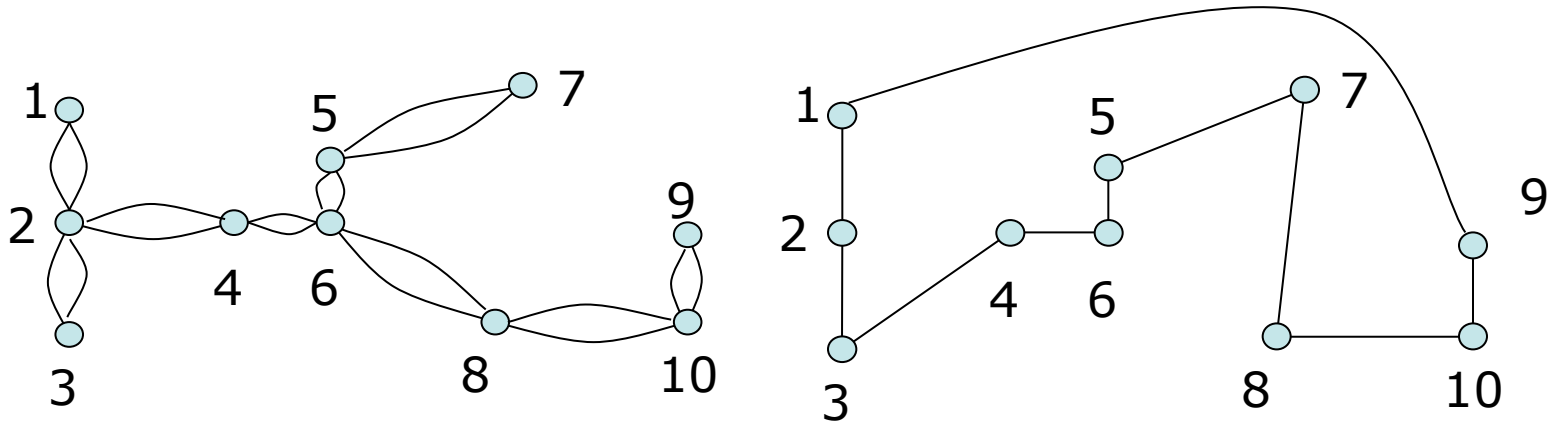
Note: W traverses each edge of T' twice: $C(W) = 2 C(T) \leq 2 \text{OPT}$

Step 4: Find a tour H by “shortcutting” W :

1, 2, 3, ~~2~~, 4, 6, 5, 7, ~~8~~, ~~6~~, 8, 10, 9, ~~10~~, ~~8~~, ~~6~~, ~~4~~, ~~2~~, 1

Final solution $H = 1, 2, 3, 4, 6, 5, 7, 8, 10, 9, 1$

Δ -TSP: A 2-approximation



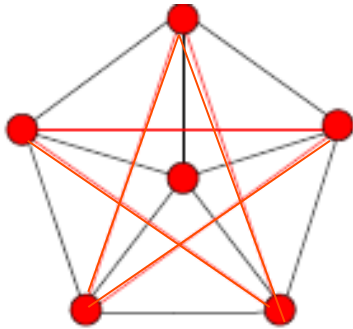
$C(H) \leq C(W)$, because of the triangle inequality

Hence: $C(H) \leq C(W) \leq 2 \text{ OPT}$

QUESTION: What is the complexity of this algorithm ?

Δ -TSP: Tightness of 2-approximation

Example

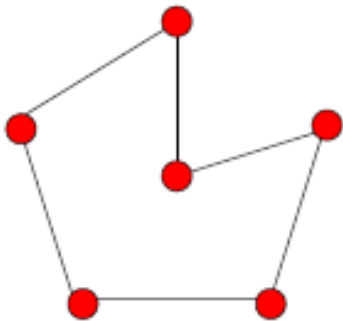


Complete graph K_n

Red edges: $w = 2$

Other edges: $w=1$ (union of a star + cycle)

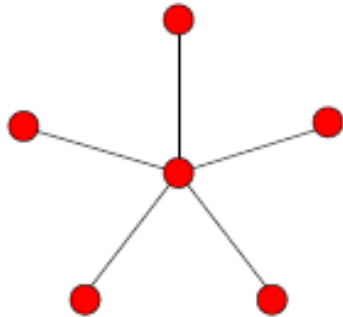
Optimal tour



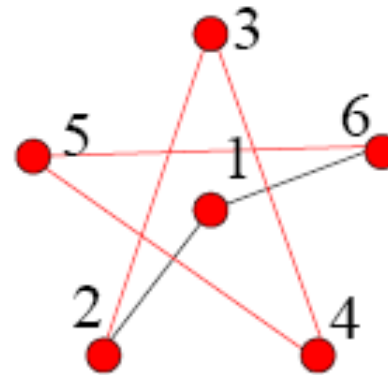
OPT = n

Δ -TSP: Tightness of 2-approximation

Minimum MST



Solution



$$C(H) = (n-2)*2 + 2*1 = 2n-2$$

$$\text{Hence, } C(H) / \text{OPT} = (2n-2) / n \rightarrow 2$$

Δ -TSP: improvement to $\rho = 1.5$

Theorem: There is a 1.5-approximation algorithm for Δ -TSP [Christofides 1976]

Step 1: Start again by finding a minimum spanning tree, T , of cost $C(T)$

- We cannot now just double the edges, this will not avoid a loss of 2
- But we would still like to create an Eulerian graph starting from T
- What makes T non-Eulerian?
- Problematic vertices: vertices of odd degree
- Claim: The number of odd-degree vertices is even (why?)

Δ -TSP: improvement to $\rho = 1.5$

Detour on matchings

Consider a graph $G = (V, E)$

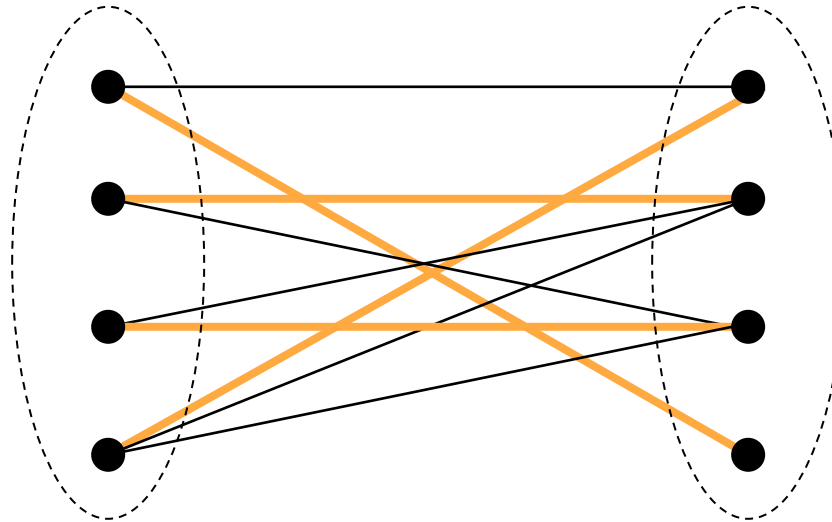
Definition: A matching M is a collection of edges $M \subseteq E$, such that no 2 edges share a common vertex

Given a matching M , a vertex u is called *matched* if there exists an edge $e \in M$ such that e has u as one of its endpoints

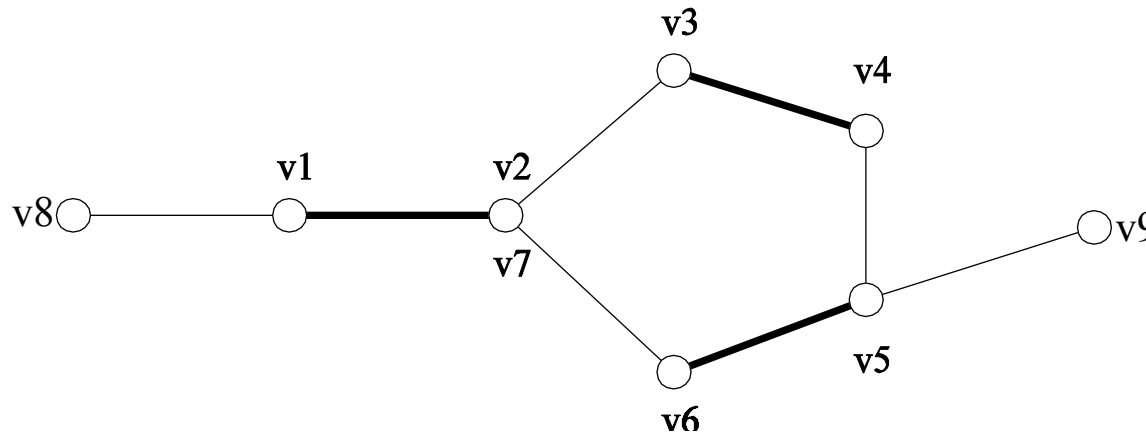
Δ -TSP: improvement to $\rho = 1.5$

Detour on matchings

Examples



a matching in a bipartite graph



A matching in general graphs (vertex $v8$ is unmatched)

Δ -TSP: improvement to $\rho = 1.5$

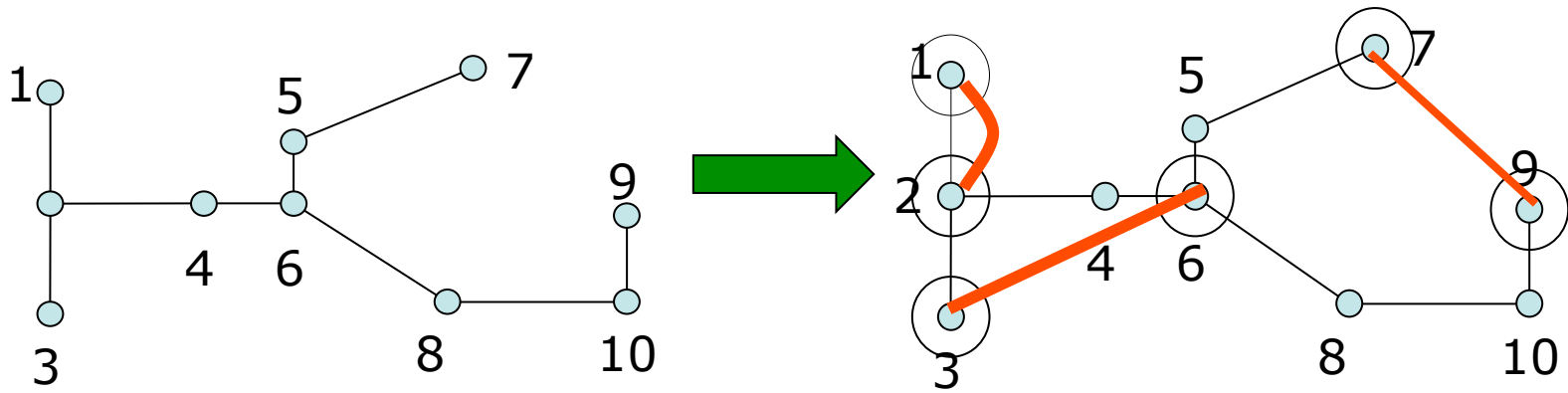
Detour on matchings

Types of matchings we are interested in:

- **Maximal matching:** find a matching where no more edges can be added
- **Maximum matching:** find a matching with maximum number of edges
- **Perfect matching:** find a matching where every vertex is matched
- **Maximum weight matching:** given a weighted graph, find a matching with maximum possible total weight
- **Minimum weight perfect matching:** given a weighted graph, find a perfect matching with minimum cost

All the above problems can be solved in polynomial time (several algorithms and publications over the last decades)

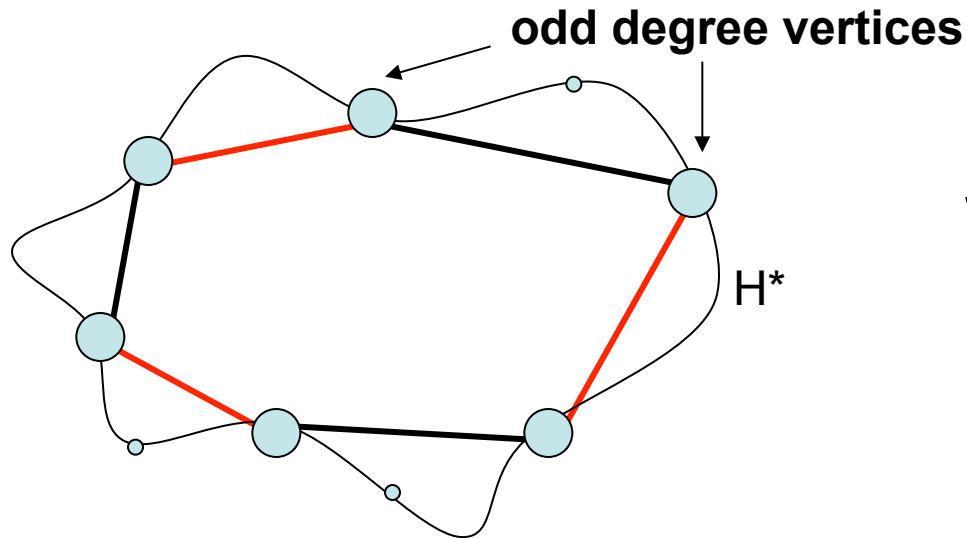
Δ -TSP: improvement to $\rho = 1.5$



Step 2:

- Find the set of vertices of T of odd degree, say S
- S contains an even number of vertices
- Consider the graph G_S induced by S
- Find a minimum weight perfect matching, M , in G_S

Δ -TSP: improvement to $\rho = 1.5$



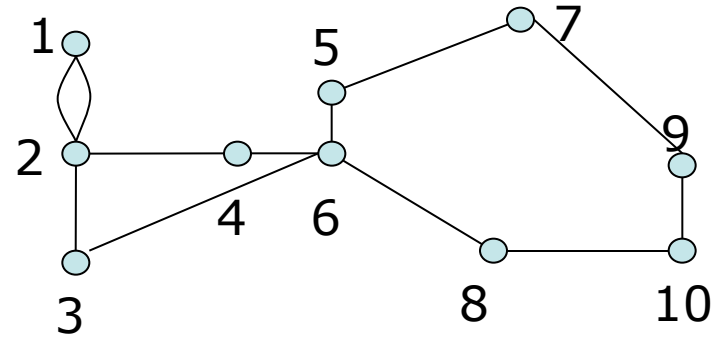
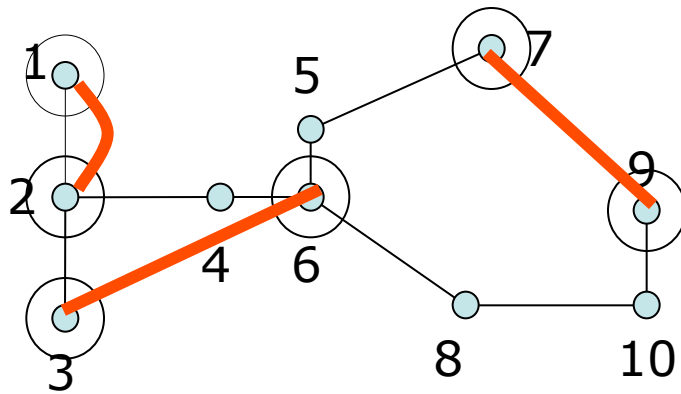
Why is a minimum cost perfect matching useful?

- Let H^* be an optimal TSP tour
- Shortcut the tour to vertices of S
- This leads to a tour over S
- By triangle inequality, cost of S -tour $\leq C(H^*) = \text{OPT}(I)$
- S -tour can be decomposed into 2 perfect matchings of S , the red (M_1), and the black (M_2)

Then $C(H^*) \geq C(M_1) + C(M_2) \geq C(M) + C(M)$, since M is optimal

Hence, $C(M) \leq C(H^*) / 2 = \text{OPT}(I)/2$

Δ -TSP: improvement to $\rho = 1.5$



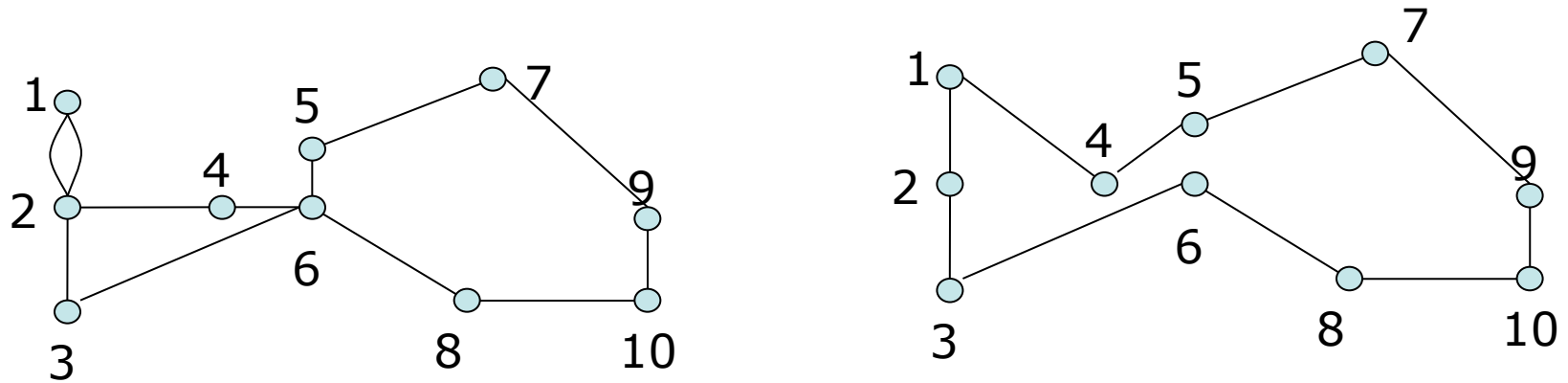
Step 3:

- Add the edges of M to T and let T' be the obtained (multi)graph
- All vertices of T' are of even degree now
- Find an Euler cycle, W, in T'

Euler cycle W: 1, 2, 3, 6, 8, 10, 9, 7, 5, 6, 4, 2, 1

$$C(W) = C(T) + C(M) \leq C(H^*) + C(H^*) / 2 = 1.5 C(H^*)$$

Δ -TSP: improvement to $\rho = 1.5$



Step 4:

Find a tour H by shortcutting W:

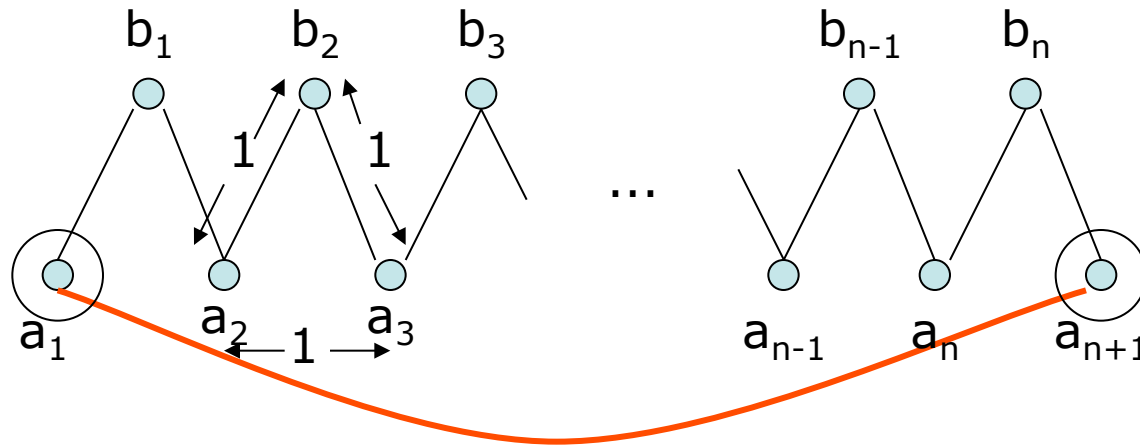
H: 1, 2, 3, 6, 8, 10, 9, 7, 5, ~~6~~, ~~4~~, ~~2~~, 1

$C(H) \leq C(W)$, by use of the triangle inequality

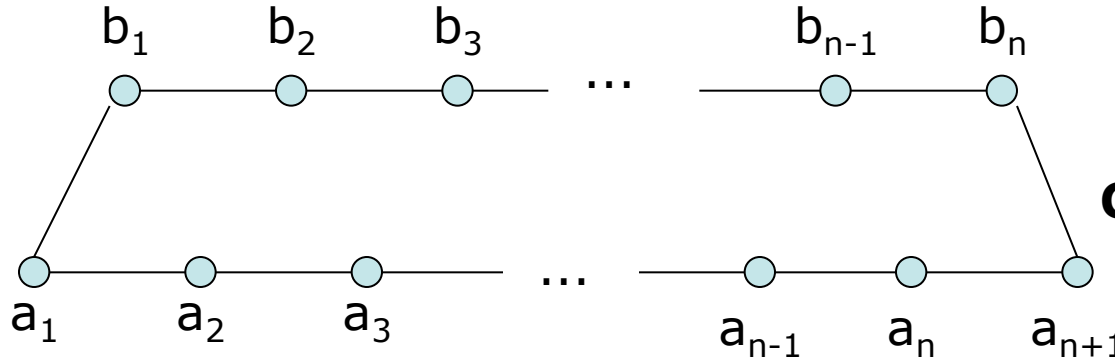
Hence, overall: $SOL(I) = C(H) \leq C(W) \leq 1.5 C(H^*) = 1.5 OPT(I)$

QUESTION: What is the complexity of this algorithm ?

Δ -TSP: Tightness of 1.5-approximation



$$C(H) = n + n + n = 3n$$



H^*

$$C(H) = n + (n-1) + 2 = 2n + 1$$

$$C(H) / C(H^*) \rightarrow 3/2$$