Travelling Salesman Problem

Anamul Hoque Emtiaj(1905113)
Sk. Saifullah Hafiz(1905114)
Nur Hossain Raton(1905117)

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

March 3, 2023

Table of contents

Table of Contents

Problem Statement

Travelling Salesman Problem

Given a set of cities and the distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point

Problem Statement

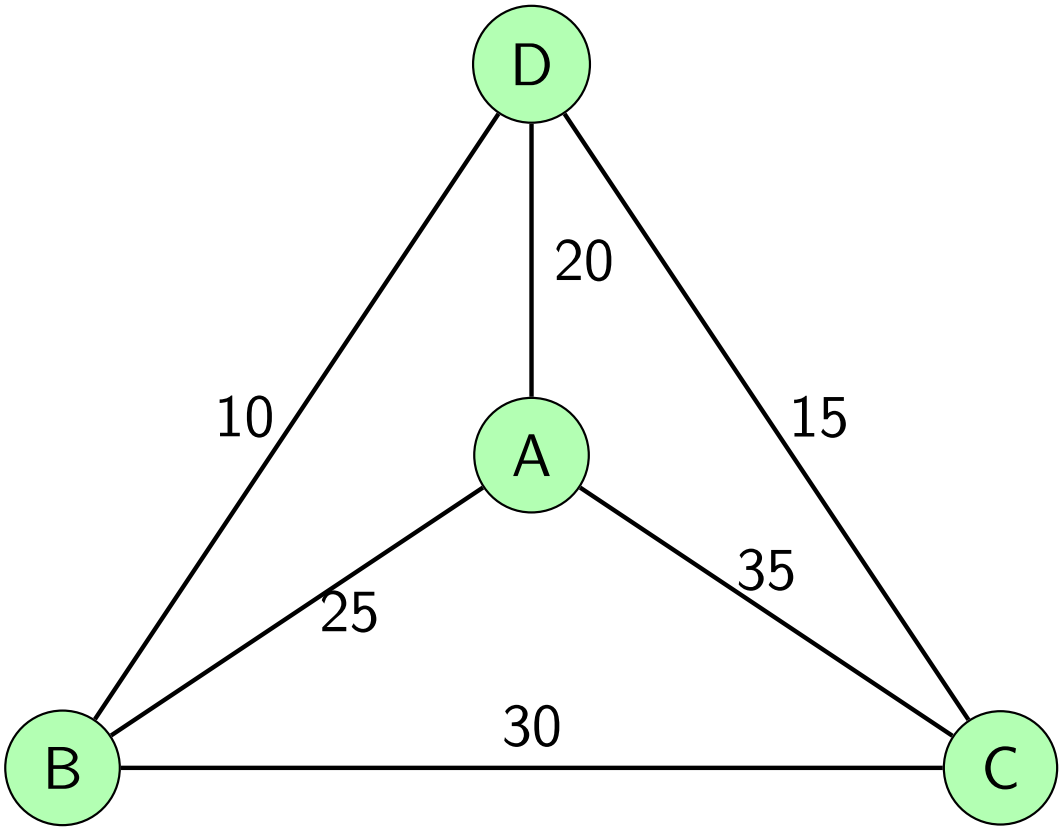


Table of Contents

Application

Genetics and Biology

- to optimize the order in which different genes are sequenced

Image : https://bit.ly/3KyXsqz

Application

Logistics and Transportation

-to optimize the delivery routes of goods, services, or people

Image : https://bit.ly/3IMfl3V

Application

Computer Wiring

-connecting together computer components using minimum wire length

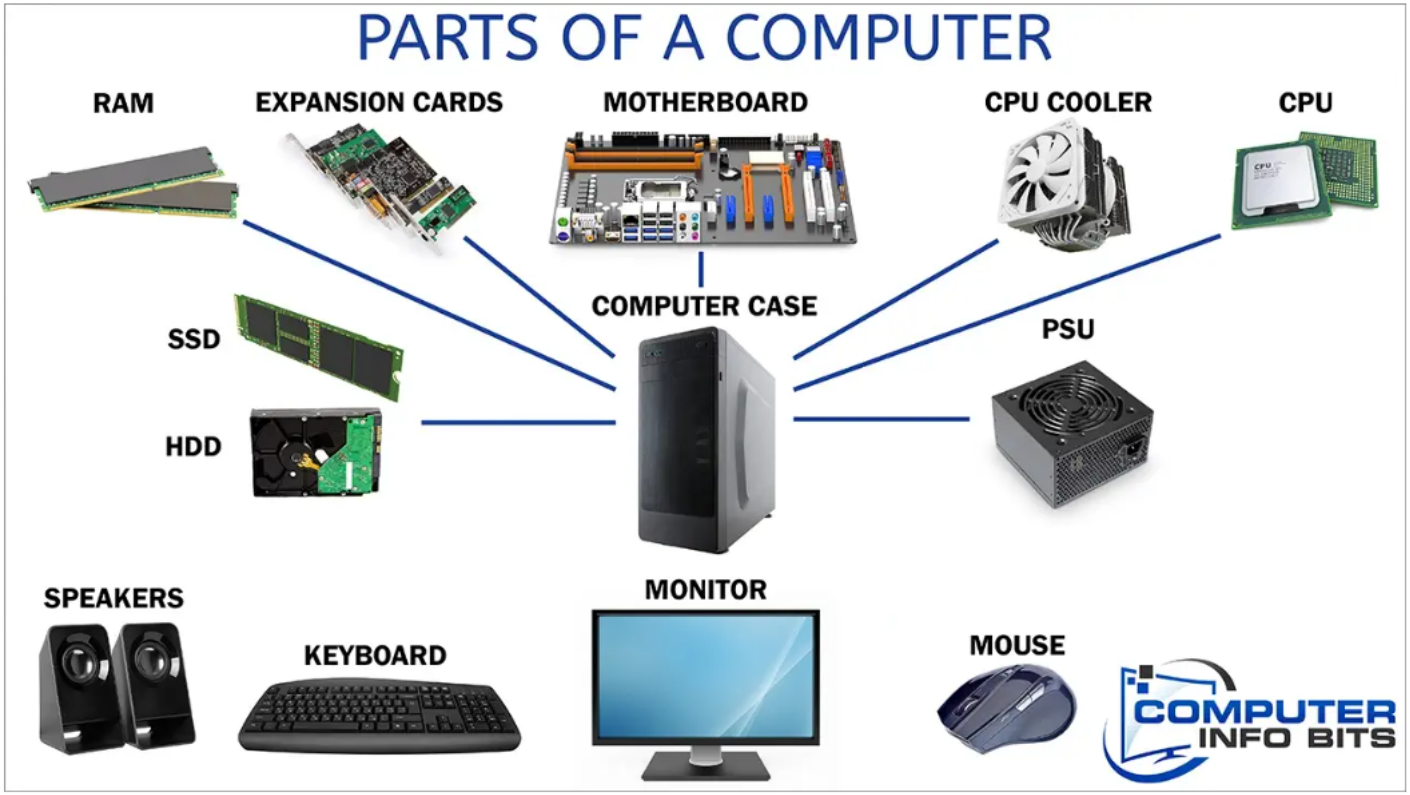


Image : https://bit.ly/3YYr44W

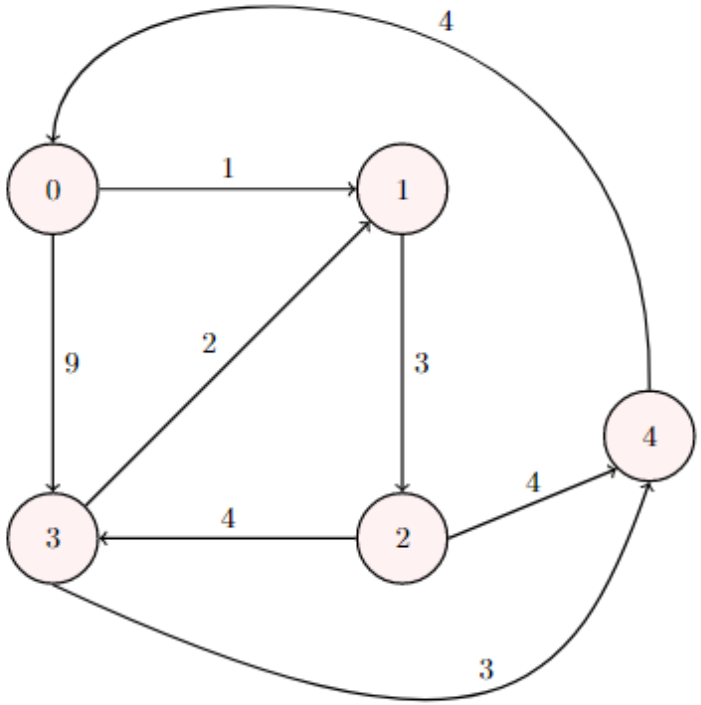
Application

- *Network Design*
- *Circuit Board Design*
- *Job Sequencing*
- *Airplane Scheduling*

And many more ...

Table of Contents

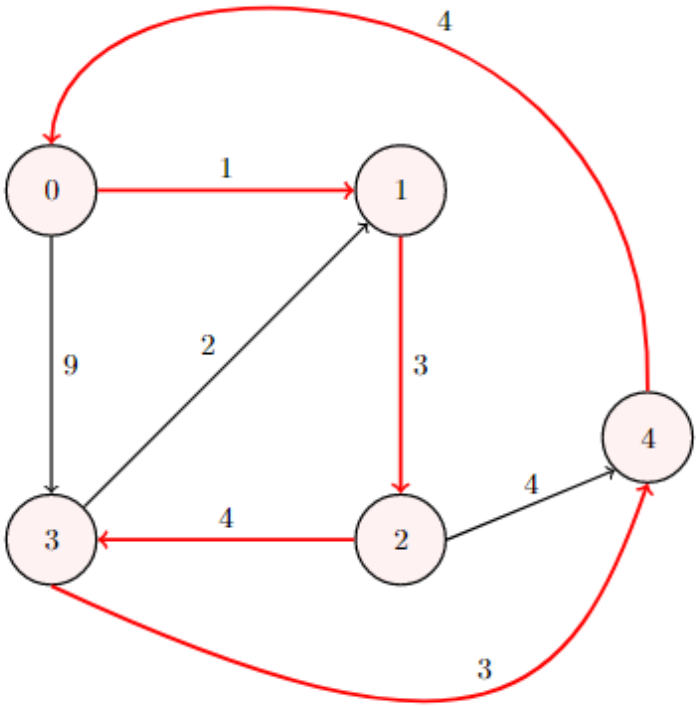
How to implement?



What will be the minimum cost?

How to implement?

Path 1:

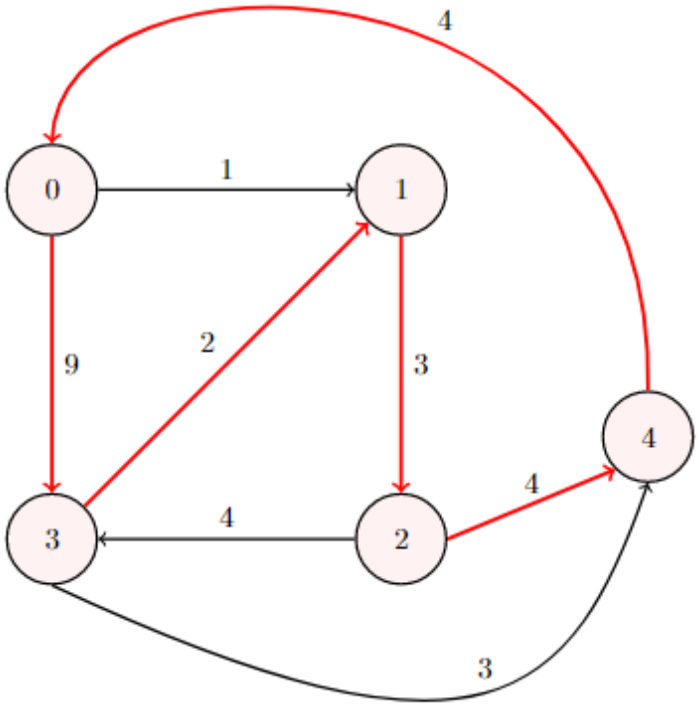


- $0 - 1 - 2 - 3 - 4 - 0$

Cost: $1(0\text{-}1) + 3(1\text{-}2) + 4(2\text{-}3) + 3(3\text{-}4) + 4(4\text{-}0) = 15$

How to implement?

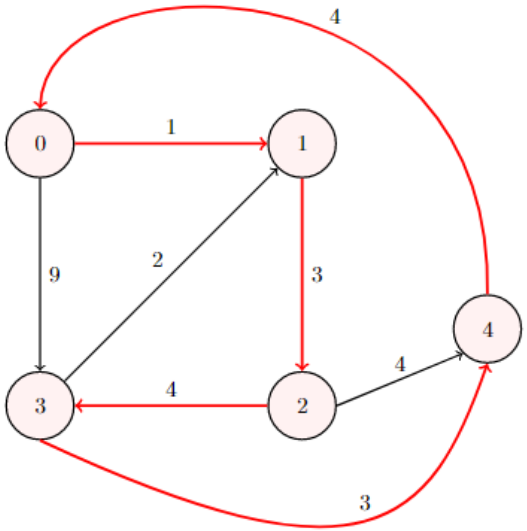
Path 2:

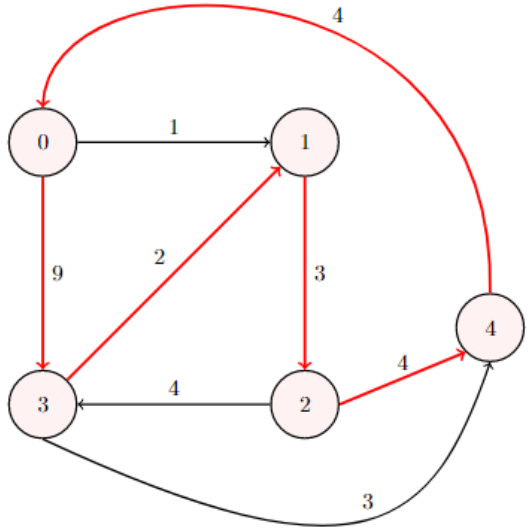


- $0 - 3 - 1 - 2 - 4 - 0$

Cost: $9(0\text{-}3) + 2(3\text{-}1) + 3(1\text{-}2) + 4(2\text{-}4) + 4(4\text{-}0) = 22$

How to implement?





Here the minimal cost is 15

- A complete graph with n vertex gives n! different path combinations
- Which requires $O(n^2 n!)$ runtime
- No Polynomial time solution
- NP-hard problem

How to implement?

- Others possible ways of solutions

How to implement?

- Others possible ways of solutions
  1. Branch and Bound

How to implement?

- Others possible ways of solutions
  1. Branch and Bound
  2. Approximation using MST

How to implement?

- Others possible ways of solutions
  1. Branch and Bound
  2. Approximation using MST
  3. Bitmask Dynamic Programming

How to implement?

- Others possible ways of solutions
  1. Branch and Bound
  2. Approximation using MST
  3. Bitmask Dynamic Programming
- Here we only see Bitmask DP algorithm

Table of Contents

Sub Problem

- *Define a function* $f(i)$

Sub Problem

- *Define a function* $f(i)$
  - Distance for exploring the rest of the cities starting from city i.

Sub Problem

- *Define a function $f(i)$*
  - Distance for exploring the rest of the cities starting from city i.
- *But what is the problems?*

Sub Problem

- *Define a function $f(i)$*
  - Distance for exploring the rest of the cities starting from city i.
- *But what is the problems?*
  - Can't travel one city more than once.

Sub Problem

- *Define a function $f(i)$*
  - Distance for exploring the rest of the cities starting from city i.
- *But what is the problems?*
  - Can't travel one city more than once.
  - Must remember which city we already visited.

Why Bit Mask?

- *Use bit mask for denoting city status*

Why Bit Mask?

- *Use bit mask for denoting city status*
- *A 32 bit integer use as mask*

Why Bit Mask?

- *Use bit mask for denoting city status*
- *A 32 bit integer use as mask*
  - Bit i indicates, whether city i is already visited or not.

Why Bit Mask?

- *Use bit mask for denoting city status*
- *A 32 bit integer use as mask*
  - Bit i indicates, whether city i is already visited or not.
    1. 0 for not visited.

Why Bit Mask?

- *Use bit mask for denoting city status*
- *A 32 bit integer use as mask*
  - Bit i indicates, whether city i is already visited or not.
    1. 0 for not visited.
    2. 1 for visited.

..Cont'd Subproblem

- *Add additional parameter mask.*

..Cont'd Subproblem

- *Add additional parameter mask.*
  - Our updated function, $f(i, mask)$.

..Cont'd Subproblem

- *Add additional parameter mask.*
  - Our updated function, $f(i, mask)$.
  - i indicates present city.

..Cont'd Subproblem

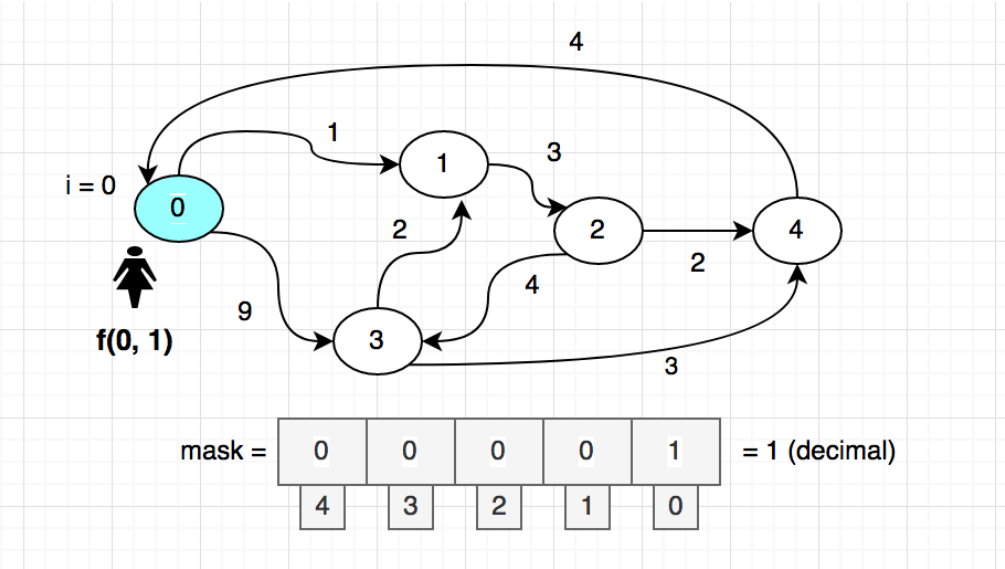
- *Add additional parameter mask.*
  - Our updated function, $f(i, mask)$.
  - i indicates present city.
  - mask contains information about the tour.
- *How do we implement bit masking?*

Bit Masking

If started from city 0.

Bit Masking

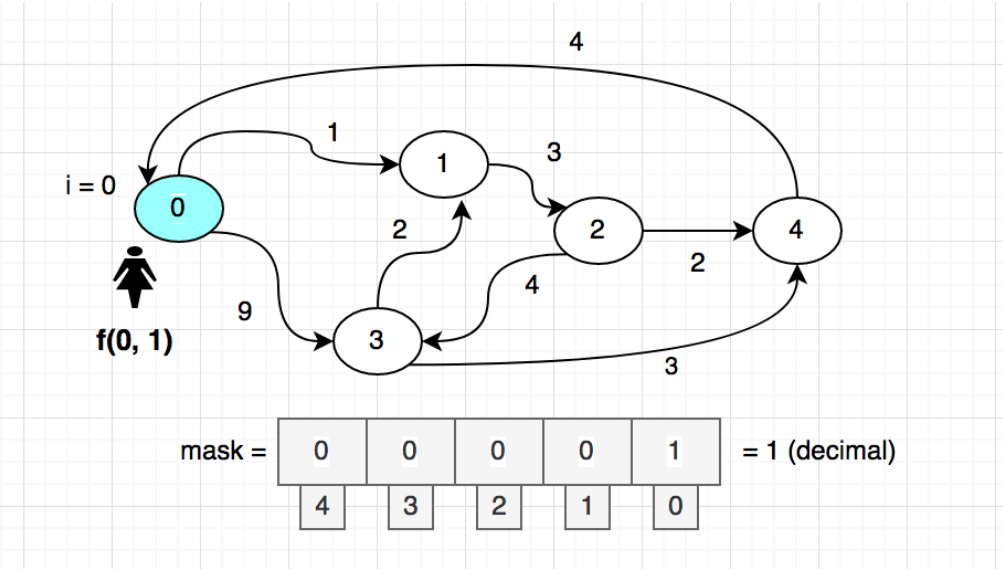
If started from city 0.



Bit Masking

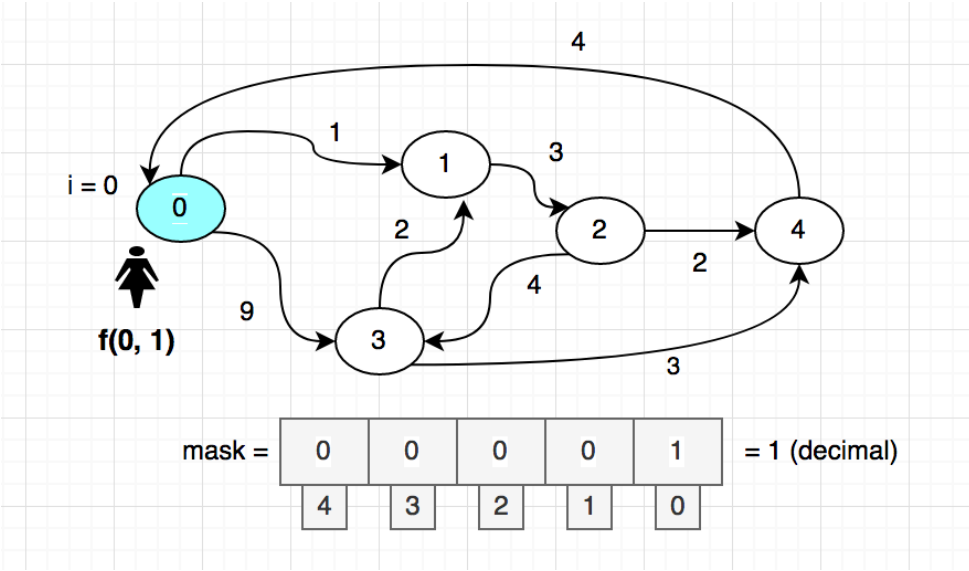
If started from city 0.

Then he goes to city 3.

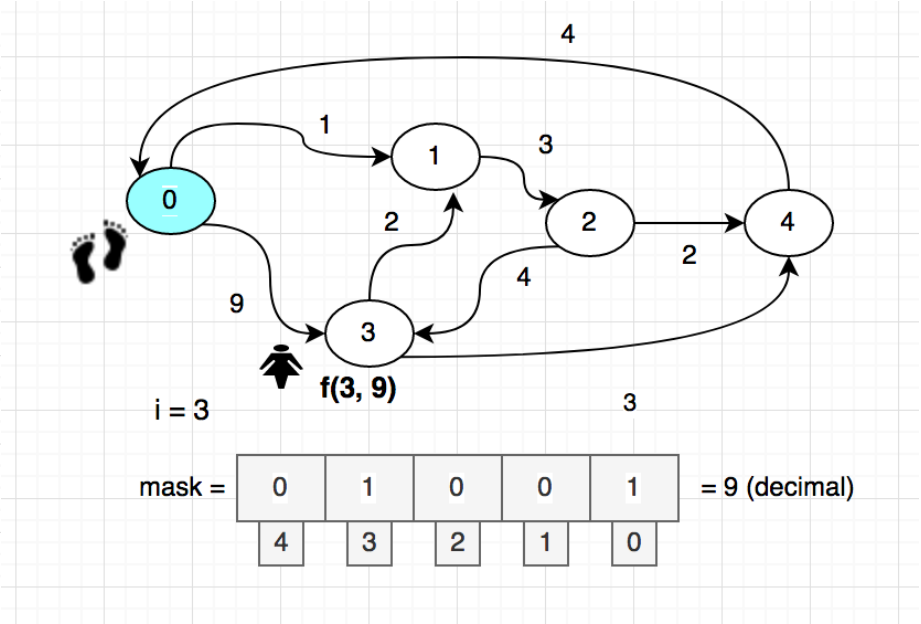


Bit Masking

If started from city 0.



Then he goes to city 3.



Bit Masking

```
int turnOn(int x,int pos)
{
    return x | (1<<pos);
}
```

Code snippet for turning on a bit

Bit Masking

```
int turnOn(int x,int pos)
{
    return x | (1<<pos);
}
```

Code snippet for turning on a bit

```
bool isOn(int N,int pos)
{
    return (bool)(N & (1<<pos));
}
```

Code snippet for checking a bit

Recursive Formula

- We have to solve $f(0, 1)$

Recursive Formula

- We have to solve $f(0, 1)$
- Base case $f(i, 2^{n-1})$

Recursive Formula

$f(i, 2^{n-1}) = dis[i][0]$
$f(i, mask) = min(f(j, turnOn(mask, j)) + w[i][j]) where i, j \in E$

Algorithm

```
#define EMPTY_VALUE -1
#define MAX_N 10
#define INF 1061109567

int w[MAX_N][MAX_N];
int mem[MAX_N][1<<MAX_N];
int n;

int f(int i, int mask)
{
    if (mask == (1<<n) - 1)
        return w[i][0];


    if (mem[i][mask] != -1)
        return mem[i][mask];


    int ans = INF;
    for (int j = 0;j < n;j++)
    {
        if (w[i][j] == INF) continue;

        if (isOn(mask,j) == 0)
        {
            int result = f(j, turnOn(mask, j)) + w[i][j];
            ans = min(ans, result);
        }
    }

    return mem[i][mask] = ans;
}
```

Time Complexity

- For n city mask value can be 0 to 2^n.

Time Complexity

- For n city mask value can be 0 to 2^n.
- So total $n * 2^n$ states possible.

Time Complexity

- For n city mask value can be 0 to 2^n.
- So total $n * 2^n$ states possible.
- Total Run time: $O(n^2 * 2^n)$

Thank you

Any Question?