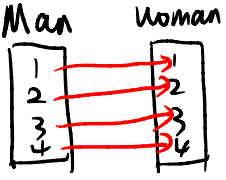
Capter 1 – stable Marriage

(a) The man optimal stable matching :

(M_1, W_1) (M_2, W_2) (M_3, W_3) (M_4, W_4)



Intermedia steps:

M_1 chooses W_1, W_1 is free then W_1 is proposed to M_1;

M_2 chooses W_2, W_2 is free then W_2 is proposed to M_2;

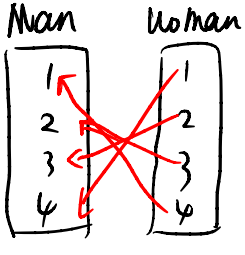
M_3 chooses W_3, W_3 is free then W_3 is proposed to M_3;

M_4 chooses W_4. W_4 is free then W_4 is proposed to M_4;

Then the progrom comes to an end.

(b) The woman optimal stable matching :

(W_1, M_4) (W_2, M_3) (W_3, M_2) (W_4, M_1)



prove the finding is the woman optimal stable matching.:

Proof. (by contradiction)

S^* : (W_1, M_4) (W_2, M_3) (W_3, M_2) (W_4, M_1) Suppose exists a certain woman with someone other than her best valid partner.

∵ Women propose in the decreasing order of preference list.

∴ Some woman must be rejected by her best valid partner.

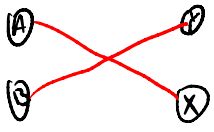
make some ascumptions. make A be the first woman,

and let X be the first valid man that rejects her,

so that X proposed to B, who he prefers to A.

let S be another stable matching where A and X are matched

And assume that Let Y be B's partner.

S: 

In S^*, B is not rejected, however A is rejected by X and A is the first woman is rejected and B follows descending order, so B prefers x to Y;

∵ X prefers B to A; B prefers X to Y

∴ B-X is unstable pair, the S is an unstable matching.

what's more, every woman gets their right man who ranks the first in their preference list, so the matching must be woman optimal one.

(c)

Man #	Preference List
1	1 2 3 4
2	1 2 3 4
3	1 2 3 4
4	1 2 3 4

Woman #	Preference List
1	4 3 2 1
2	3 4 1 2
3	2 1 4 3
4	1 2 3 4

Step 1:
$M_1 \rightarrow W_1$, W_1 accepted

Step 2:
$M_2 \rightarrow W_1$, W_1 rejected M_1, W_1 is proposed to M_2

Step 3:
$M_3 \rightarrow W_1$, $W_1 \leftrightarrow M_3$

Step 4:
$M_4 \rightarrow W_1$, $M_3 \not\leftrightarrow W_1$ $M_4 \leftrightarrow W_1$

Step 5:
$M_1 \rightarrow W_2$, $M_1 \leftrightarrow W_2$

The total steps are 10.

Step 6:
$M_2 \rightarrow W_2$, W_2 rejected M_2

Step 7:
$M_2 \rightarrow W_3$, $W_3 \leftrightarrow M_2$

Step 8:
$M_3 \rightarrow W_2$, $W_2 \not\leftrightarrow M_1$, $M_3 \leftrightarrow W_2$

Step 9:
$M_1 \rightarrow W_3$ W_3 rejected M_1

Step 10:
$M_1 \rightarrow W_4$, $W_4 \leftrightarrow M_1$

(d) Truthfulness in Stable Marriage.

If lying exists, it is likely to make it possible for men and women to end up better off.

The true preference list:

	Preference List		
w_3	m_2	m_1	m_3
w_2	m_1	m_2	m_3
w_1	m_1	m_2	m_3

	Preference List		
m_1	w_3	w_2	w_1
m_2	w_2	w_3	w_1
m_3	w_3	w_2	w_1

The Result is $m_1 - w_3$, $m_2 - w_2$, $m_3 - w_1$

The fake preference list:

lying part.

	Preference List		
w_3	m_2	m_3	m_1
w_2	m_1	m_2	m_3
w_1	m_1	m_2	m_3

	Preference List		
m_1	w_3	w_2	w_1
m_2	w_2	w_3	w_1
m_3	w_3	w_2	w_1

The Result is $m_2 \leftrightarrow w_3$, $m_3 \leftrightarrow w_1$, $m_1 \leftrightarrow w_2$.

∴, w_3 through changing the pair (m_3, m_1), realising a better partner.

Chapter 2 — Running Time Analysis

(a)

$g_1(n) = 2^{\sqrt{\log n}}$ $g_2(n) = 2^n$ $g_3(n) = n(\log n)^3$ $g_4(n) = n^{\frac{4}{3}}$ $g_5(n) = n^{\log n}$

$g_6(n) = 2^{2^n}$ $g_7(n) = 2^{n^2}$

The ascending order of the growth rate:

①

$g_1(n)$ comes before $g_3(n)$

$g_3(n) = 2^{\log n + 3\log(\log n)}$

$g_1(n) = 2^{\sqrt{\log n}}$

$\because \log n + 3\log(\log n) > \log n$

$\therefore O(g_3(n)) = g_1(n)$

② $g_3(n)$ comes before $g_4(n)$

$g_4(n) = n \cdot n^{\frac{1}{3}}$, we need to judge the relationship between $n^{\frac{1}{3}}$ and $(\log n)^3$ it is easy to tell $O(n^{\frac{1}{9}}) = \log n$

$\therefore O(n^{\frac{1}{3}}) = (\log n)^3$

③ $g_4(n)$ comes before $g_5(n)$ it is obvious $O(\log n) = \frac{4}{3}$

④ $g_5(n)$ comes before $g_2(n)$ $g_5(n) = 2^{\log(n^{\log n})}$

$= 2^{(\log n)^2}$

it is obvious that $O(n) = (\log n)^2$

⑤ $g_2(n)$ comes before $g_7(n)$ $\quad O(n^2) = n$

⑥ $g_7(n)$ comes before $g_6(n)$ To compare 2^n with n^2, it is obvious that polynomials grow slower than exponentials.

The final result: $g_1 \rightarrow g_3 \rightarrow g_4 \rightarrow g_5 \rightarrow g_2 \rightarrow g_7 \rightarrow g_6$

(b)

(1) $f(n) = n^3$, for the outer loop to traverse the range of i, it costs n iterations. for the inner loop to traverse the range of j, it cost almost n iterations. For the sum from A[i] through A[j] takes $O(j-i+1)$ operations, which is always at most $O(n)$. Hence the total time comlexity is $O(n^3)$

(2) we only choose part of all the calculation to verify it's $\Omega(n^3)$.

we choose $i \leq \frac{1}{3}n$ and $j \geq \frac{2}{3}n$ which meets that all the $j > i$, and the least calculation time for a certain pair (i, j) is $(\frac{2}{3}n - \frac{1}{3}n) + 1 > \frac{1}{3}n$

Hence the total time expenditure for imcomplete calculation is $(\frac{1}{3})^n \times \frac{1}{3} = \frac{n^3}{27}$, hence the algorithm is $\Omega(n^3)$.

Another provement.

we can Calculate the exact time complexity of traverse.

Maths Analyze

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n} j-i+1$$

$$= \int_1^n \int_{x+1}^{n} y-x+1 \, dy \, dx$$

$$= \int_1^n \left[\frac{1}{2}y^2-(x-1)y\right]\Big|_{x+1}^{n} dx$$

$$= \int_1^n \left[\frac{n^2}{2}-(x-1)n-\frac{(x+1)^2}{2}+(x^2-1)\right] dx$$

$$= \int_1^n \left[\frac{x^2}{2}-(n+1)x+\frac{n^2}{2}+n-\frac{3}{2}\right] dx$$

$$= \int_1^n \left[\frac{x^2}{2}-(n+1)x+\frac{n^2+2n-3}{2}\right] dx$$

$$= \frac{1}{6}x^3-\frac{n+1}{2}x^2+\frac{n^2+2n-3}{2}x\Big|_1^n$$

$$= \left(\frac{n^3}{6}-\frac{n^3+n}{2}+\frac{n^3+2n^2-3n}{2}\right)-\left(\frac{1}{6}-\frac{n+1}{2}+\frac{n^2+2n-3}{2}\right)$$

$$= \left(\frac{n^3}{6}+n^2-2n\right)-\left(\frac{1}{6}-\frac{n+1}{2}+\frac{n^2+2n-3}{2}\right)$$

$$= \frac{n^3}{6}+\frac{n^2}{2}-\frac{5}{2}n+\frac{11}{6}$$

the result is both $O(n^3)$ and $\Omega(n^3)$

(3)

my algorithm

```
C[0]=0
for i = 1, 2, ··· n do
    C[i]=C[i-1]+A[i]
end for

for i=1, 2, ··· n do
    for j=i+1, i+2 ··· n do
        B[i,j] = C[j]-C[i-1]
    end for
end for
```

we need an array to store the sum of prefix number.

to traverse the first loop, it costs n iterations.

for the double loop, the outer loop costs n iterations, and the inner loop costs at most n iterations.

The sum time cost is $n^2+n \leq n^2+n^2 = 2n^2$

so the f(n) (designed algorithm) $= O(n^2)$, which is more efficient than that listed in previous paragraph.

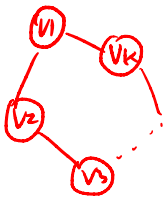
Chapter 3 — Graphs

(a) from the question, the graph $G=(V,E)$ is connected so the m (edge numbers) must more than $n-1$ ($n \rightarrow$ vertex numbers) $m \geq n-1$. And only when $m=n-1$, the G_1 turns to a tree. So $G=T$ means that the graph G contains $n-1$ edges and is a tree.

proof by contradiction.

we assume G is not a tree and is undirected and connected. G must contain a cycle A.

we assume A contain k vertexs. from $V_1 \rightarrow V_2 \rightarrow \cdots \rightarrow V_k \rightarrow V_1$ and those vertexs form a circle.



For the breadth-first search tree T, if we start from V_1, the edge V_1-V_2 and the edge V_1-V_k will all covered in the tree.

However, for the deepth-first search tree T', if we start from V_1, and V_1 first traverses V_2. And V_2 must reaches to V_k because the two vertexs are connected, so the edge V_1-V_k will not be included in T'.

So the $T' \neq T$, it proofs that G must be a tree.

Now that G is a tree and T is a tree which contains all the edges on G, so $G = T$.

(b)

i.

we elicit the connected component T including node s and node t. And the vertex number of the connected component T is n', $n' \leq n$.

Consider any breadth-first tree P of T with s as the root.

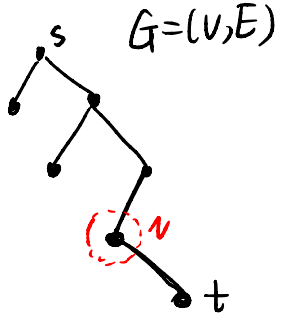
Because each node v has a level is the shortest distance to s.

the t has a least level of $l = \frac{n}{2} + 1$

if as for all the levels from 1 to $\frac{n}{2}$, randomly deleting a node from those levels will not breaking the connectivity, there at least are two nodes at a random level.

So that the total vertex will add up at least to $2 \cdot \frac{n}{2} + 2 = n+2$ (including s and t). $\because n+2 > n \geq n'$, $\therefore$ there must be a level in range 1 through $\frac{n}{2}$ containing just one node z, as long as deleting z, the connectivity will be destructed.

ii.



$n=7$

$s \to t$ distance is 4

if delete vertex v, the connectivity will be destructed.

iii.

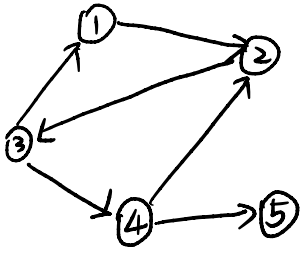
1. construct a breadth-first search Tree with the start with vertex s. During the tree-building stage, for every level we construct a $Li[\,]$, containing all the nodes on the i-th level.

2. we traverse the $list_1 \cdots list_{\frac{n}{2}}$, to find the $list_2$, in $list_2$ there is only one node and then give the right node v in $list_2$.

Step 1's time complexity is $O(m+n)$; Step 2's time complexity is $O(n)$; so the total complexity is $O(m+n)$ time.

(c)

Turn the adjacency list into actual graph like:

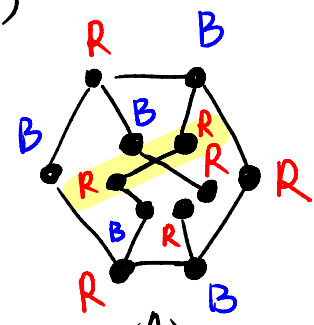


Because this is a directed graph, and the outdegree of the 5-th node is 0, it's impossible to start the tour from the 5-th node.

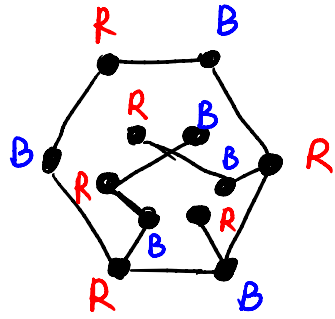
Besides, if we start the visiting from another node (except 5-th node), it's impossible to go back from 5-th node, so we can never reach the start node again.

From every node, it is impossible for us to do the special tour.

(d)



(A)



(B)

(A) is not a bipartite. There exists two points with the same color.

(B) doesn't contain any edge with ends of the same color.

B is a bipartite.

(e)

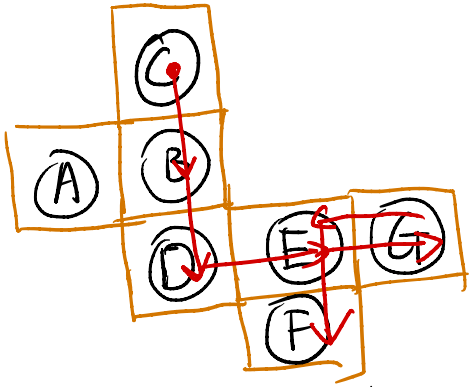
i. Design the algorithm based on depth-first search.

we put the room into coordinate.

```
void Traverse (int x, int y)
{  if (x. y is the destination) keep the robot still, return.
   for i=1, ... 4 do
      if i=1
         if no obstacles upwards and target position isn't reached
            move upwards
            Traverse(x, y+1)
            move downwards
```

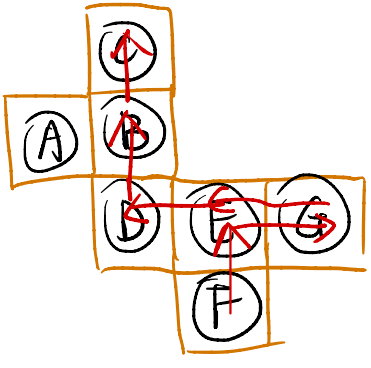
```
if i=2
    if no obstacles on the right and target position isn't reached
        move to the right
        Traverse (x+1, y)
        move to the left

if i=3
    if no obstacles downwards and target position isn't reached
        move downwards
        Traverse (x, y-1)
        move upwards

else
    if no obstacles on the left and target position isn't reached
        move to the left
        Traverse (x-1, y)
        move to the right.

end for.
}
```

ii)



The route of my algorithm is C→B→D→E→G →E→F

iii)



the route of the back travel is F→E→G→E →D→B→C

the distance has to cover is 6 steps.