Team ZerOne
Computer & Networks Engineers

ملخص

خوارزميات

عضو الجمعية الطلابية :

عبدالرحمن أبوزيد

قال عليه السلام :
{ من كان في حاجة أخيه كان الله في حاجته }

Scanned by CamScanner

"Algorithms"

Chapter 1 : Algorithms Efficincy, Analysis and Order.

- what is the algorithms :- step by step to solve a problem.

- Some important words :-

1. problem :- Question.
2. Solution :- answer.
3. parameter :- The variables in the problem.
4. instance :- The value for the parameters.

شرح بالعربي :-

1. في الخوارزميات لا نطلق على السؤال Question بل نطلق عليه problem.
2. في الخوارزميات لا نطلق على الجواب answer بل نطلق عليه Solution.
3. parameters :- هي المتغيرات الموجودة في الـ problem "السؤال".
4. instance :- هي القيم البدائية للمتغيرات في السؤال.

Scanned by CamScanner

Example :-

1 sort a list(S) of (n) numbers in nondecreasing order if S = [10, 7, 11, 5, 13, 8].
what is the solution, parameter, and instance for this problem.

* solution :-

S = [5, 7, 8, 10, 11, 13]

* parameter :-

$\boxed{S}$, $\boxed{n}$

* instance :-

القيم المعطاة في السؤال ⇐

S = [10, 7, 11, 5, 13, 8]

n = 6

هو عدد القيم في (S)

Scanned by CamScanner

Example :-

[2] Determine whether the number (x) is in the list (S) of (n) numbers. The answer is yes if (x) is in (S) and no if it is not, Also determine the paramete & instance if S = [10, 7, 11, 5, 13, 8] and x = 5.

* solution :-

"yes"

* parameters :-

S , n , x

* instance :-

S = [10, 7, 11, 5, 13, 8]
x = 5
n = 6

ملاحظة :-

الهدف هو عمل ما يهدف إلى الحل في أسرع وقت ممكن ، وبالتالي في Example (2) أفضل من Example (1) والسبب في (1) يجب المرور على جميع العناصر ولكن في (2) لا يشترط المرور على جميع العناصر حتى وإن كان هناك في الفرضية التي تؤديه الخوارزمية .

Scanned by CamScanner

Some examples of algorithms :-

1. Sequential search.

⇐ ويقصد بها البحث عن عنصر معين.

2. Add array members.

⇐ ويقصد بها جمع عناصر مصفوفة.

3. Exchange sort.

⇐ تبديل ما بين قيمتين

4. Matrix multiplication.

⇐ حاصل ضرب 2 matrix

5. Binary search.

⇐ البحث عن رقم معين بالنسبة للمنتصف إما في الجزء العلوي أو السفلي.

Scanned by CamScanner

ZERO _One_Team

⇐ ملاحظة قبل الشرح: في الإمتحان لا يطلب منك كتابة كود معين
هي فقط الفهم.

⇐ ملاحظة أخرى: معرفة ما المقصود بـ Basic operation.

Basic operation :- we call the instruction or group of instructions in the algorithm.
هي جملة أو مجموعة جمل تحدد المطلوب في الخوارزمية.

[1] Sequential search :-

⇒ problem: Is the key x = 4 in the array S=[3,10,4,7] of(n)key?

⇒ what is the parameter, and instance in the problem?

```
void seqsearch (int n,
                const keytype S[ ],
                key type x,
                index & location)
{ location = 1;
  while (location <= n && S[location] != x)
    location++;
  if (location > n)
      location = 0;
```

* solution :-
  "yes"

* parameter :-
  x , S , n

* instance :-
  x = 4
  S = [3,10,4,7]
  n = 4

Scanned by CamScanner

2 Add Array Members :-

⇒ problem : add all the number in the array S = [7, 3, 10, 2, 6] of (n) numbers ?

⇒ what is the parameter, instance and Basic Operation?

```
number sum (int n , const number S[ ])
    index i ,
    number result ;
    result = 0 ,
    for (i = 1 , i <= n , i ++)
        result = result + S[i] ;
    return result
```

⇒ The Basic operation
الجملة المبني عليها الجواب.

* Solution :-

result = 28

* parameter :-

S , n

* instance :-

S = [7, 3, 10, 2, 6]

n = 5

Scanned by CamScanner

3 Exchange sort :-

⇒ problem : sort n keys in nondecreasing order for S = | 10 | 1 | 3 | 2 | 5 |

⇒ what is the parameters, instance and the Basic Operation.

```
exchangesort (int n, keytype S[])
    index i, j ;
    for (i=1 , i<=n; i++)
        for (j=i+1; j<=n ; j++)
            if (S[j] < S[i])
                exchange S[i] and S[j] ;
```

⇒ The Basic operation

* solution :-

10	1	3	2	5

⇓

1	10	3	2	5

⇓

1	3	10	2	5

⇓

1	2	10	3	5

⇓

1	2	3	10	5

⇓

1	2	3	5	10

solution

* Parameter :-

S , n

* instance :-

S = [10, 1, 3, 2, 5]

n = 5

Scanned by CamScanner

[4] Matrix multiplication :-

⇒ problem: Determine the two product A x B matrix when $A = \begin{bmatrix} 1 & -1 \\ 2 & 0 \end{bmatrix}$, $B = \begin{bmatrix} 1 & 1 \\ 3 & 2 \end{bmatrix}$

⇒ Determine the Basic operation, parameter, and instance.

```
void matrixmult (int n, const number A[][]
                 const number B[][], const number C[][])
index i, j, k;
for (i=1; i<=n; j++)
    for (j=1, j<=n, j++)
    C[i][j] = 0;
    for (k=1, k<=n; k++)
C[i][j] = C[i][j] + A[i][k] * B[k][j];
}
}
```

⇒ Basic Operation

* solution :-

$$C = \begin{bmatrix} -2 & -1 \\ 2 & 2 \end{bmatrix}$$

* parameter :-

A , B

* instance :-

$$A = \begin{bmatrix} 1 & -1 \\ 2 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 1 \\ 3 & 2 \end{bmatrix}$$

Scanned by CamScanner

5 Binary search :-

⇒ problem: Determine whether (x)=15 is in sorted array S = {-1, 1, 3, 15, 20} of (n) keys.

⇒ Determine the parameter and the instance of the problem?

```
void binsearch(int n, const keytype S[],
                keytype x, index &location)
  index low, high, mid;
  low=1; high=n;
  while (low <= high && location==0){
  mid = L(low+high)/2];
  if (x==S[mid])
    location=mid;
  else if (x<S[mid])
    high=mid-1;
  else
    low=mid+1; }
```

⊙ ملاحظة:-
أفضل من Binary search
Sequential search والسبب في
الـ Binary لا نمر على جميع العناصر
مما يعمل على اختصار الوقت.

* Solution :-

-1	1	3	15	20

↳ mid

* solution :-
"yes"

* parameter :-
x , S , n

* instance :-
x = 15
S = {-1, 1, 3, 15, 20}
n = 5

Scanned by CamScanner

The Fibonacci Sequence

↳ It is another example for algorithem.

* في هذا النوع من الخوارزمية يعتمد حل أي عنصر على حل وجواب العنصر الذي قبله.

* Fibonacci sequence يوجد منه نوعان

1. Recursive
2. Iterative

* توضيح كيف أن كل عنصر يعتمد على العنصر الذي قبله

$f_0 = 0$

$f_1 = 1$

$f_2 = f_1 + f_0 = 1 + 0 = 1$

$f_3 = f_2 + f_1 = 1 + 1 = 2$

$f_4 = f_3 + f_2 = 2 + 1 = 3$

$f_5 = f_4 + f_3 = 3 + 2 = 5$

$f_6 = f_5 + f_4 = 5 + 3 = 8$

⋮

etc

Scanned by CamScanner

*توضيح بسيط لحالة recursive و Iterative :-

*ملاحظة :- الكودات ليست للحفظ فقط للتوضيح .

1 Recursive	2 Iterative:

fib(int n)
if (n<=1)
return n;
else
return fib(n-1)+fib(n-2);
	int fib2(int n)
	{ index i;
	int f[0...n];
	f[0]=0;
	if (n<0)
	f[1]=1;
	for(i=2,i<=n;i++)
	f[i]=f[i-1]+f[i-2];
	} return f[n]; }

⊙ في حالة Iterative أفضل من حالة Recursive بالرغم من أنّ كلاهما يحلان نفس المبدأ والسبب :-
حلقة (for) الموجودة في الـ Iterative تحل جميع العناصر المطلوبة دون الاستدعاء في كل مرة كما يحدث في Recursive فتقلل الوقت وهذا هدف الخوارزميات .

→ (decreasing Time)

Scanned by CamScanner

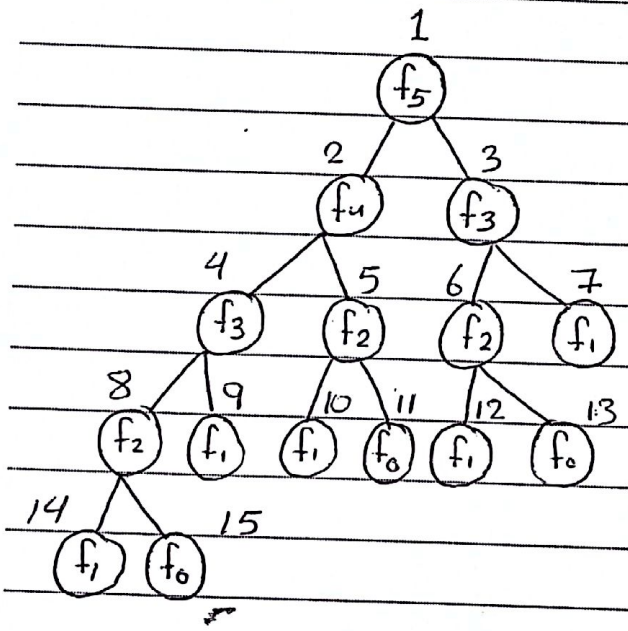
⊙ في الامتحان يأتي السؤال كالتالي :-

⇒ what is the number of terms computed?

⇐ معنى هذا السؤال كم عدد العناصر التي يعتمد عليها عنصر معين .

Example :-

⇒ what is the number of terms for f_5 ?



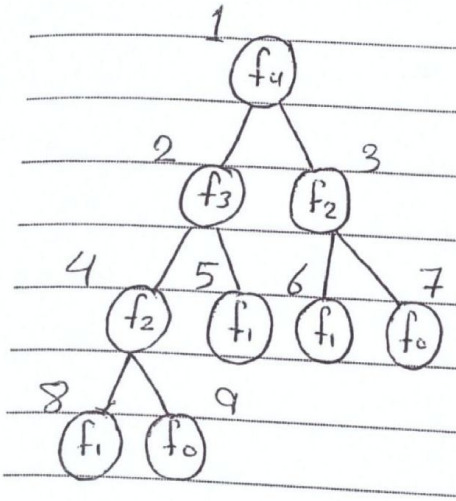
The number of terms for f_5 is $\boxed{15}$

* لحل السؤال ارسم مخطط الشجرة وفقط عد العناصر التي تحتويه .

Scanned by CamScanner

Example :-

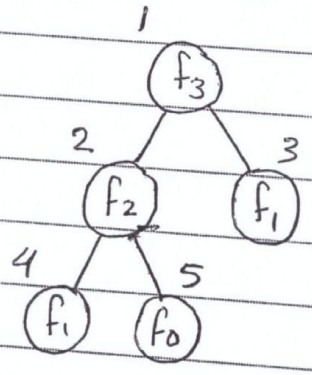
⇒ what is the number of terms for f_4 ?



* The number of terms for f_4 is $\boxed{9}$

Example :-

⇒ what is the number of terms for f_3 ?



* The number of terms for f_3 is $\boxed{5}$

Scanned by CamScanner

Complexity Analysis

Time complexity Analysis: how many times the basic operation is done for each value.

Time complexity analysis يستخدم لمعرفة كم مرة يتم تكرير الـ Basic operation.

لحساب Time complexity يجب معرفة أن بعض الخوارزميات في الحل يمر على جميع العناصر في المصفوفة وبعض الخوارزميات لا يضطر للمرور على جميع العناصر.

الخوارزمية التي تمر على جميع العناصر يكون لها "Every case".

الخوارزمية التي لا تمر على جميع العناصر لها 3 حالات:

1. Best case.
2. Averege case.
3. Worest case.

⇐ وبالتالي الخوارزمية إما أن تكون لها حالة Every case "أو" Best case
Average case
worest case

Examples على الخوارزميات

1. sequential search.
2. Add array members.
3. Exchange sort.
4. matrix multiplication.
5. Binary search.

سوف نرى أن sequential search و Binary search لا تمر على جميع العناصر.

بينما add array members و exchange sort و matrix multiplication تمر على جميع العناصر.

لها every case

لها 3 حالات
1. Best case.
2. Averege case.
3. worest case.

تعريفات مهمة :-

* Every case :- when this is operation is always done the same number of times for every instance of size(n) $\Rightarrow T(n)$

← تمر على جميع عناصر المعروفة.

* Worest case :- The maximum number of times the algorithim will ever do its basic operation for size n $\Rightarrow W(n)$

← يكون العنصر المطلوب آخر عنصر فيمر على جميع العناصر

Scanned by CamScanner

* Average case :- the average of the number of times the algorithm dose the basic operation for an input size of (n) ⇒ A(n).

إلى عندما يكون العنصر المطلوب في المنتصف.

* Best case :- The minimum number of times the algorithm will ever do its basic operation for an input size of n ⇒ B(n).

إلى عندما يكون العنصر المطلوب هو أول عنصر.

⇒ what is the (every case) for ?

1. Add array member :-

"فقط"

$$T(n) = n$$

2. Exchange sort :-

$$T(n) = \frac{n(n-1)}{2}$$

3. Matrix multiplication :-

$$T(n) = n^3$$

Scanned by CamScanner

⇒ Sequential search has [1] Best case [2] Worest case
[3] Averege case ⇒ وال بيدا قرى كل نقيع العناصر

Sequential search :-

[1] Best case :- "حفظ"

$$B(n) = 1$$

[2] Worest case :-

$$W(n) = n$$

[3] Average case :-

$$A(n) = \frac{n+1}{2}$$

"أي نسبة هنا مثلا"

في الإمتحان إذا قال لك 70% نسبة عدم الوجود لتحسب الـ averegecase
تعوض في هذا القانون مكان الـ (P)

$$A(n) = n\left(1 - \frac{P}{2}\right) + \frac{P}{2}$$

Scanned by CamScanner

Order

complexity categories :-

$\Theta(\lg n)$ $\Theta(n)$ $\Theta(n \lg n)$ $\Theta(n^2)$ $\Theta(n^3)$ $\Theta(2^n)$ $\Theta(e^n)$ $\Theta(n!)$

الأفضل ← الأفضل ← الأسوء

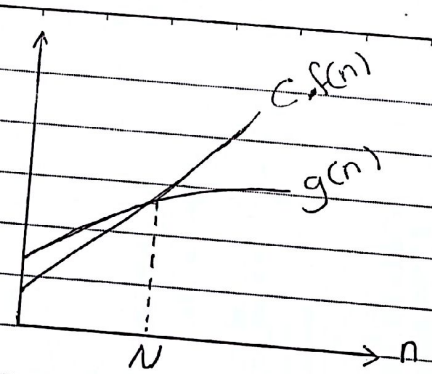
⊙ شرح ما المقصود بـ big Θ و big O و big Ω :-

big O :- The previous discussion imparted an intuitive feel for order (Θ), Here we developing theory that enables us to define order rigorously.

→ difintion :- For given complexty function f(n), O(f(n)) is the set of complexity function g(n) for which there exists some positive real constant (C) and some nonnegative integer N such that for all $n \leq N$

$$g(n) \leq C \times f(n)$$

Scanned by CamScanner



$g(n) \leq C \times f(n)$; $\boxed{n \geq N}$

ملاحظة :-

we say that "big O" puts an asymptotic $\boxed{\text{upper bound}}$ on function.

Example :-

سيتم التوضيح أكثر مع الأمثلة :-

show that $\overbrace{n^2}^{g(n)} \in O(\overbrace{n^2+10n}^{f(n)})$. Because, for $n \geq 0$.

طريقة الحل تأخذ صورة $\boxed{1}$ $C = n$ على الأقل صورة واحدة يجب أن يكون

$\boxed{2}$ $C > n$ الجواب صحيح.

$\boxed{3}$ $C < n$

$\boxed{1}$ $C = 0$, $n = 0$

$0 \leq 0$ (✓)

$\boxed{2}$ $C = 2$, $n = 1$

$1 \leq 22$ (✓)

$\boxed{3}$ $C = 1$, $n = 2$

$4 \leq 44$ (✓)

$\therefore n^2 \in (n^2 + 10n)$

$g(n) = n^2$

$f(n) = n^2 + 10^n$

الأرقام موضحة في القانون

$g(n) \leq C \times f(n)$

Scanned by CamScanner

Example :-

show that $n \in O(n^2)$, Because, for $n \geq 1$

→ Solution :-

$g(n) = n$ $\quad g(n) \leq C \times f(n)$

$f(n) = n^2$

$\boxed{1}$ $C = 1$, $n = 1$

$1 \leq 1$ (✓)

ملاحظة :-
بالنسبة للأرقام بإمكانك اختيار
الرقم الذي تريده .

$\boxed{2}$ $C = 2$, $n = 1$

$1 \leq 2$ (✓)

ملاحظة : ليس بالضرورة أن
تكون جميع الحالات (✓)
على الأقل حالة واحدة .

$\boxed{3}$ $C = 1$, $n = 2$

$2 \leq 4$ (✓)

$\therefore n \in O(n^2)$

Scanned by CamScanner

Example :-

show that $n^2 + 10n \in O(n^2)$. Because, for $n \geq 1$

⇒ solution :-

$g(n) = n^2 + 10n$ $\quad g(n) \leq C \times f(n)$

$f(n) = n^2$

1 $C = 1$, $n = 1$

$11 \leq 1$ (X)

2 $C = 15$, $n = 1$

$11 \leq 15$ (✓) ⇐ على الأقل وحدة صح حتى يكون proving صحيح.

3 $C = 1$, $n = 2$

$14 \leq 4$ (X)

$\therefore n^2 + 10n \in O(n^2)$

الطريقة المختصرة :- / يأتي في الامتحان

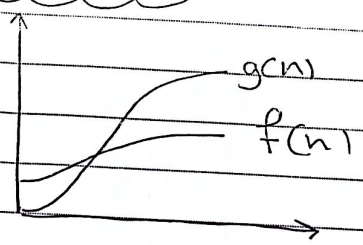
انظر الى رتبة $g(n)$ و $f(n)$ إذا كانت رتبة $f(n)$ أكبر من $g(n)$ فهي تنتمي إلى big O

ex : $g(n) = n^{(1)}$

$f(n) = n^{(2)}$ $\quad g(n) \leq C \times f(n)$

Scanned by CamScanner

big Ω



The rule :- $g(n) \geq C \times f(n)$.

example :-

prove that

$n^3 \geq n^2$

بدون عمل الـ proving

$g(n) = n^3$

$f(n) = n^2$

رتبة $g(n)$ أكبر من رتبة $f(n)$

وبالتالي $g(n) \geq C f(n)$

$\therefore n^3 \geq n^2$

Scanned by CamScanner

big Θ

$\rightarrow$ $c \cdot f(n) \leq g(n) \leq d \cdot f(n)$

باختصار رتبة $f(n)$ يجب أن تساوي $g(n)$

ex: $g(n) = n^3 + 2$
$f(n) = n^3 + n$

نفس الرتبة من الدرجة الثالثة:
وبالتالي تنتمي إلى big Θ

ex:
$g(n) = n^4$
$f(n) = n^4 + n$

نفس الرتبة / big Θ

ex: $g(n) = n^4$
$f(n) = n^2 + 2$
big Ω

ex: $g(n) = n^2 + 2$
$f(n) = n^4$
big O

Scanned by CamScanner

$$n^2 + 10n \in \Theta(n^2) \quad \text{and} \quad n^2 \in \Theta(n^2 + 10n).$$

This means that Θ separates complexity functions into disjoint sets. We will call these sets ***complexity categories.*** Any function from a given category can represent the category. For convenience, we ordinarily represent a category by its simplest member. The previous complexity category is represented by $\Theta(n^2)$.

The time complexities of some algorithms do not increase with n For example, recall that the best-case time complexity $B\ n$ for Algorithm 1.1 is 1 for every value of n. The complexity category containing such functions can be represented by any constant, and for simplicity we represent it by Θ (1).

The following are some important properties of order that make it easy to determine the orders of many complexity functions. They are stated without proof. The proofs of some will be derived in the exercises, whereas the proofs of others follow from results obtained in the next subsection. The second result we have already discussed. It is included here for completeness.

Properties of Order:

1. $g(n) \in O(fn))$ if and only if $f(n) \in \Omega(g(n))$.
2. $g(n \in \Theta(f(n))$ if and only if $f(n \in \Theta(g(n))$.
3. If $b > 1$ and $a > 1$, then $\log_a n \in \Theta(\log_b n)$.

$b > a > 0$
$a^n \in o(b^n)$

This implies that all logarithmic complexity functions are in the same complexity category. We will represent this category by $\Theta(\lg n)$.

4. If $b > a > 0$, then

   $a^n \in o(b^n)$.

   $g(n) \leq f(n)$

   This implies that all exponential complexity functions are not in the same complexity category.

5. For all $a > 0$

   $a^n \in o(n!)$.

   This implies that $n!$ is *worse* than any exponential complexity function.

6. Consider the following ordering of complexity categories:

   $$\boxed{\Theta(\lg n) \quad \Theta(n) \quad \Theta(n \lg n) \quad \Theta(n^2) \quad \Theta(n^j) \quad \Theta(n^k) \quad \Theta(a^n) \quad \Theta(b^n) \quad \Theta(n!)}$$

   where $k > j$ and $b > a > 1$. If a complexity function $g(n)$, is in a category that is to the left of the category containing $f(n)$, then

   $g(n) \in o(f(n))$.

7. If $c \geq 0$, $d > 0$, $g(n)$, $\in O(f(n))$, and $h(n) \in \Theta(f(n))$, then

Scanned by CamScanner

Chapter 2: Divide and Conquer

* In this Unit we will talk about 3 examples for devide & conquer:-

1. Binary search.
2. Meregesort.
3. Quick sort.

⇒ Firstly you should know that :

Divide is : Divid an instance of a problem recursively into two or more smaller instance until the solutions to the small instance are obtainable.

Conquer is : The solution.

⟹

Scanned by CamScanner

Binary search :-

↳ It is, to search for a key x in sorted array.

example 1 :-

suppose x = 18 and we have the following array :-

10 12 13 14 18 20 25 27 30 35 40 45 47

توضيح

⇐ لحل هذا السؤال في الـ Binary search :-

نأخذ العدد في المنتصف ونبدأ التقسيم ونكرر هذه العملية في كل قسم نقسمه إلى أن نصل إلى X .

في حال عدم وجود X فهذا يعني أنها غير موجودة .

The solution →

solution for example 1 :-

10 12 13 14 18 20 25 27 30 35 40 45 47

المنتصف

10 12 13 14 18 20	27 30 35 40 45 47

middle

10 12	14 18 20

The solution

سوف نختار الجزء الأيسر لنبدأ في تقسيمه والسبب X = 18 وهي أصغر من 25 .

يمكن اختيار 13 أو 14 ليكون المنتصف هنا ليس مهم والسبب عدد الأرقام عدد زوجي .

نختار الجزء الأيمن والسبب X = 18 وهي أكبر من 13

example 2 :- suppose x = 31

2 7 12 13 22 23 27 30 31 35 40 45 47

2 7 12 13 22 23

30 31 35 40 45 47

30 31 35

45 47

The solution

* Binary search ~~to~~ works by 3 steps :-

1. Divide :- The array into two subarrays about half as large. If x is smaller than the middle item, choose the left subarray. If x is larger than the middle item, choose the right subarray.

2. Conquer (solve): the subarray by determining whether x is in that subarray. Unless the subarray is sufficiently small, use recursion to do this.

3. Obtain :- The solution to the array from the solution to the subarray.

Scanned by CamScanner

* Worest case Time Complexity :-

⇒ Time complexity :-

$$\begin{cases} W(n) = W(\frac{n}{2}) + 1 & , \text{for } n > 1 \quad n \text{ apower of } 2 \\ W(1) = 1 \end{cases}$$

- Solution :- $W(n) = \lg n + 1$ ← هذا هو الأساس "حفظ"
- If n is not apower of 2 : $W(n) = \lfloor \lg n \rfloor + 1 \in \Theta(n)$

" or "

$\in \Theta(\lg n)$

⊙ في الامتحان إذا م ا أكتب/ي ال worest case ذكتب :-

$W(n) = \lg n + 1$

* Basic Operation for Binary search :-

⇒ The comparison of x with S[mid] .

Scanned by CamScanner

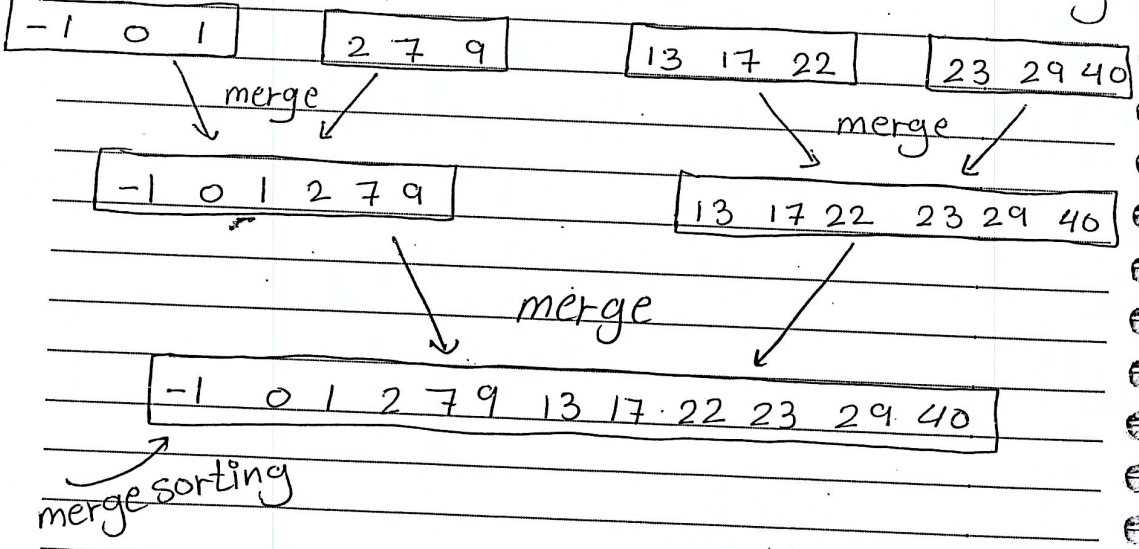
2 Mergesort :-

→ sort n keys in a sequence nondecreasing sorting.

قبل الشرح يجب أن تميز بين merge و mergesorting حيث أنّ لكل منهم worest case تختلف عن الأخرى.

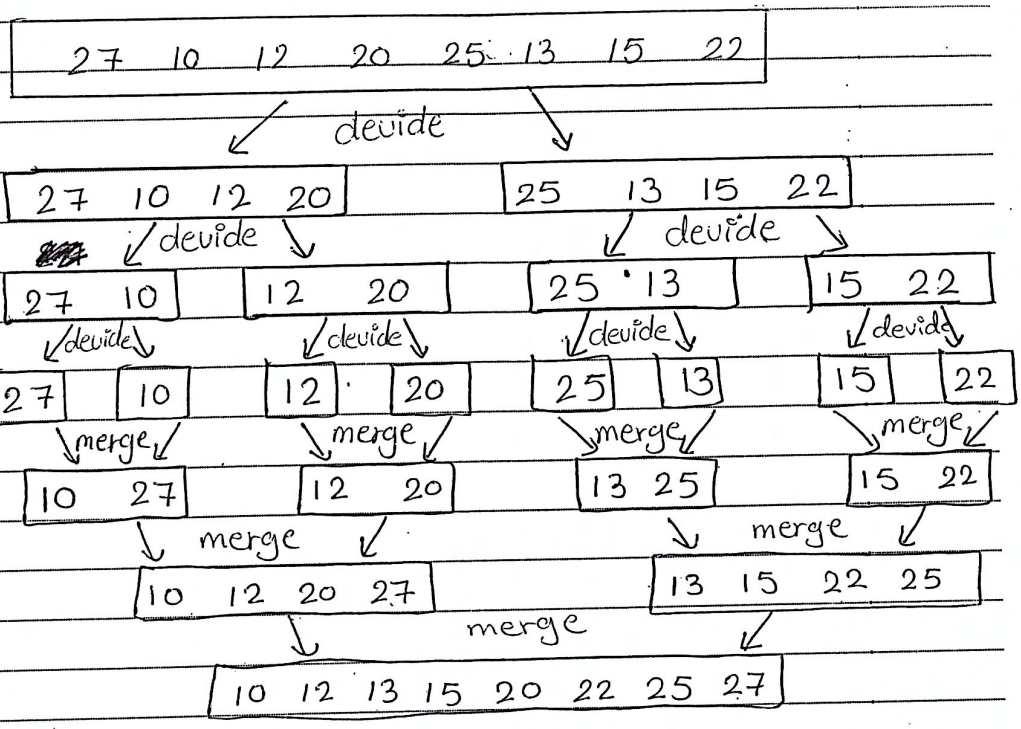
* mergesorting : sort n keys in nondecreasing sorting.
وهو الجواب النهائي عندما تكون جميع المصفوفة مرتبة.

* merge : merge two sorted array into one sorted array.



merge : # هي العملية نفسها وليس الجواب النهائي

example :-



merge sorting

طريقة الحل :-

① نبدأ بالتقسيم إلى قسمين متساويين ونبدأ بتقسيم القسمين وهكذا إلى أن يصبح كل عنصر في خانة لوحده وتعرف هذه بالـ "Devide"

② نبدأ بتجميع العناصر مع بعضهم ويشترط تجميعهم بالترتيب إلى أن تتجمع جميع العناصر بالترتيب في خانة واحدة.

Scanned by CamScanner

* merge sorting works by 3 steps :-

1. Divide the array into two subarrays each with n/2 items

2. Conqure (solve) :- each subarray by sorting it. Unless the array is sufficiently small; use recursion to do this.

3. Combine :- The solution to the subarray by merging them into a signal sorted array.

* The worst case time complexity (Merge) :-

$$W(n, m) = n + m - 1$$

* The Basic Operation for (Merge) :-

⇒ The comparison of $U[i]$ with $V[j]$.

* The worst cas time complexity (Merge sorting) :-

$$W(n) = n\lg n - (n-1) \in \Theta(n \lg n)$$

* The Basic Operation for (Merge sorting) :-

⇒ The comparison that takes place in merge.

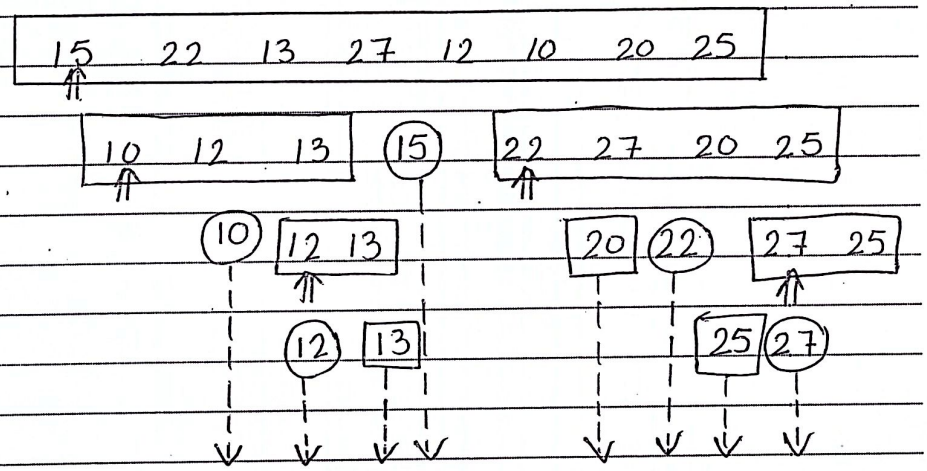
Scanned by CamScanner

13 Quick sort :-

↳ sort n keys in a sequence nondecreasing sort.

Quick sort has the same goal with merge sort but in deffrent way.

example 1 :-



The solution ⟹ 10 12 13 15 20 22 25 27

⟹ شرح طريقة الحل :-

طريقة الحل:-

① نختار رقم وهو ما يعرف باسم "pivot" وهو عنصر نقسم العناصر على أساسه ما إذا كان أكبر أو أصغر.

② بالعادة يكون أول رقم في المصفوفة هو الـ "pivot".

③ الشرط هو أن جميع تقسيم اللاحق يجب أن يكون نفس موقع الـ "pivot" الذي اخترته أي أنه إذا كان أول عنصر جميع التقسيمات التي تليه سيكون الـ "pivot" أول شيء وهنا أفضل شيء.

④ تستمر بالتقسيم إلى أن يصبح كل عنصر لوحده وسوف تحصل على الجواب.

Quick sort works by 3 steps:-

1. Choose the pivot item (usully the first item).

2. Divide the array into two partitions by placing all items smaller than some pivot item befour that item and all items larger than the pivot item after it.

3. Sort each partition recursively.

Scanned by CamScanner

في Quick sort قد يكون هناك مشكلة وهي أن ال "pivot" الذي هو أول عنصر مثلاً هو أكبر عنصر في المصفوفة فتأتي كل العناصر على يساره أو أصغر عنصر فتأتي كل العناصر على يمينه فلا نستفيد شيء

Worest case time complexity (Quick sort) :-

$$T(n) = \frac{n(n-1)}{2} \in \Theta(n^2)$$ "حفظ"

Average case time complexity (Quick sort) :-

$$A(n) = 1.38(n+1)\lg n \in \Theta(n \lg n)$$

Every case time complexity (Quick sort) :-

$$T(n) = n-1 \in \Theta(n)$$

" When not to use divide and conquer "

- An instance of size n is devided into two or more instance each almost of size n ⇒ e.g. Fibonacci sequence.
- An instance of size n is devided into almost n instance of size n/c, where c is a constant.

important

جاء سؤال في الإمتحان: "فاينل"

أن هناك طالب له علامات مبعثرة غير مرتبة ترتيب تصاعدي ويريد منك أن ترتبها ثم تبحث عن عنصر "علامة" معينة.

بالإضافة إلى كتابة Time complexity للطرق التي استخدمتها في Devide & congur :-

Solution:- [1] By using merege sort [or] Quick sort

in my solution I will use Quick sort

[2] By using Binary search ← لمعرفة العلامة هل هي موجودة

*يمكنك اختيار أي طريقة تريدها:-

ex: $\boxed{X = 19}$

7 20 13 2 8 19 18

① using Quick sort

$\boxed{2}$ (7) $\boxed{(20)\ 13\ 8\ 19\ 18}$

$\boxed{(18)\ 19\ 8\ 13}$ (20)

$\boxed{(13)\ 8}$ (18) $\boxed{19}$

$\boxed{8}$ (13)

$\boxed{2\ 7\ 8\ 13\ 18\ 19\ 20}$

The array become sorted array.

تكملة الحل

$\boxed{X = 19}$

2 7 8 13 18 19 20

$\boxed{2 \quad 7 \quad 8}$ $\boxed{18 \quad 19 \quad 20}$

The solution

Time complexity for Quick sort:-

① worest case :- $T(n) = n(n-1)/2 \in \Theta(n^2)$

② Every case :- $T(n) = n-1 \in \Theta(n)$

③ Average case :- $A(n) = 1.38(n+1)\lg n \in \Theta(n \lg n)$

worest case for Binary search :- $\lg n + 1 \in \Theta(n)$

The End for Chapter 2 ^_^

Scanned by CamScanner

Chapter 3 : " Dynamic programming "

when we couldn't solve the algorithem by using devide and Conger such as (Febonacci) we will use Dynamic programming.

In this unite we will use two ways to solve the algorithem.

[1] Binomial coefficient

[2] Dynamic programming

To understand these two ways we will put an example, but befour that we will talk about them generaly.

Scanned by CamScanner

[1] The binomial coefficient :-

↳ using Devide and Conquer.

- Definition :-

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad \text{for } 0 \leq k \leq n$$

- Recursive definition :-

$$\binom{n}{k} = \begin{cases} \binom{n-1}{k-1} + \binom{n-1}{k} & , 0 < k < n \\ 1 & , k = 0 \text{ or } k = n \end{cases}$$

example 1 :-

$$\binom{4}{2} = B[4][2] = B[3][1] + B[3][2]$$
$$= B[2][0] + B[2][1] + B[2][1] + B[2][2]$$
$$= 1 + B[1][0] + B[1][1] + B[1][0] + B[1][1] + 1$$
$$= 1 + 1 + 1 + 1 + 1 + 1$$
$$= \boxed{6}$$

شرح طريقة الحل في الخلف ⟵

شرح طريقة الحل :-

حيث أن (4) هي n و (2) هي k ⇐ $\binom{n}{k} = \binom{4}{2}$

نحول المسألة الطريقة لتسهيل أسلوب الحل

$B[n][k] = B[4][2]$

قانون الحل هكذا :-

$B[n][k] = B[n-1][k-1] + B[n-1][k]$

نطرح (1) من n — نطرح (1) من k — نطرح (1) من n — تبقى k كما هي

نكرر هذا الأسلوب إلى أن تصبح $n = k$ أو $k = 0$

لتصبح 1

$B[4][2] = B[3][1] + B[3][2]$

$= B[2][0] + B[2][1] + B[2][1] + B[2][2]$

نكرر العملية كل واحدة على حداه

$k=0$ (1) — $B[1][0] + B[1][1]$ — $B[1][0] + B[1][1]$ — $n=k$ (1)

$k=0$ (1) — $n=k$ (1) — $k=0$ (1) — $n=k$ (1)

$1 + 1 + 1 + 1$

"ناتج" $\boxed{6}$

example 2 :-

$$\binom{3}{2} = B[3][2] = B[2][1] + B[2][2]$$
$$= B[1][0] + B[1][1] + 1$$
$$= 1 + 1 + 1$$
$$= \boxed{3}$$

example 3 :-

$$\binom{4}{3} = B[4][3] = B[3][2] + B[3][3]$$
$$= B[2][1] + B[2][2] + 1$$
$$= B[1][0] + B[1][1] + 1 + 1$$
$$= 1 + 1 + 1 + 1$$
$$= \boxed{4}$$

example 4 :-

$$\binom{3}{1} = B[3][1] = B[2][0] + B[2][1]$$
$$= 1 + B[1][0] + B[1][1]$$
$$= 1 + 1 + 1$$
$$= \boxed{3}$$

Scanned by CamScanner

2 Using Dynamic programming :-

الفكرة هو أننا سوف نقوم بوضعها في جدول :-

example :-

$\binom{n}{k} = \binom{3}{2} =$

	0	1	2	3	k
0	1				
1	1	1			
2	1	2	1		
3	1	3	3	1	
n					

$B[2][1] = B[1][0] + B[1][1] = 1 + 1 = \boxed{2}$

2 من الجدول

$B[3][1] = B[2][0] + B[2][1] = 1 + 2 = \boxed{3}$

$B[3][2] = B[2][1] + B[2][2] = 2 + 1 = \boxed{3}$

remember :- $\boxed{K < n}$

وهذا السبب في أن
بعض الفراغات في الجدول لم نعبئها.

example 2 :-

$B[4][2] = ?$

$= \boxed{6}$

	0	1	2	3	4	k
0	1					
1	1	1				
2	1	2	1			
3	1	3	3	1		
4	1	4	6	4	1	

n

$B[2][1] = B[1][0] + B[1][1] = 1 + 1 = 2$

$B[3][1] = B[2][0] + B[2][1] = 1 + 2 = 3$

$B[3][2] = B[2][1] + B[2][2] = 2 + 1 = 3$

$B[4][1] = B[3][0] + B[3][1] = 1 + 3 = 4$

$B[4][2] = B[3][1] + B[3][2] = 3 + 3 = 6$

$B[4][3] = B[3][2] + B[3][3] = 3 + 1 = 4$

* Time complexity :-

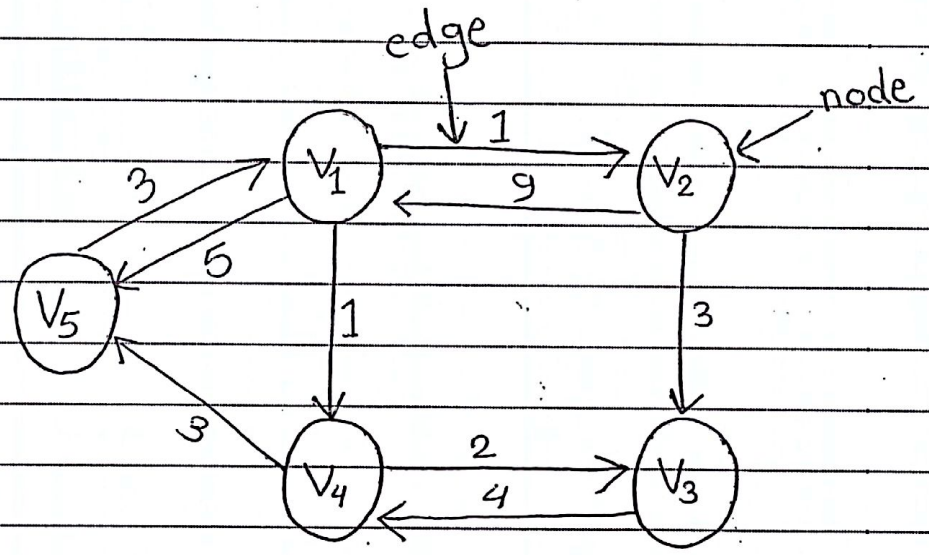
$$\frac{(2n-k+2)(k+1)}{2} \in (nk)$$

Scanned by CamScanner

" Floyd's Algorithm for shortest path "

* Terminology :-

1. graph : vertices , edges .
2. directed graph .
3. weighted graph .



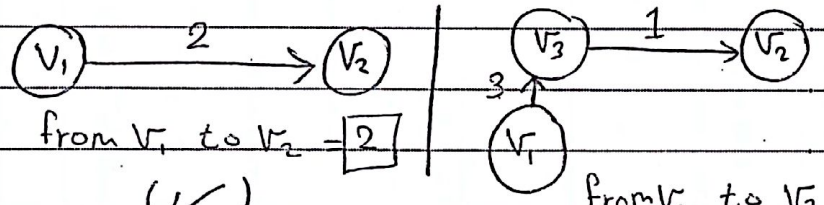
Scanned by CamScanner

* To solve Floyd's Algorithm for shortest path we will use 2 ways [1] W ⇒ wight graph [2] D ⇒ dynamic graph

* في الإمتحان هو بحدد لك بأي طريقة لحل السؤال ولكن في العادة بيتم أو بيطلب منك حل السؤال بطريقة W .

[1] طريقة W :

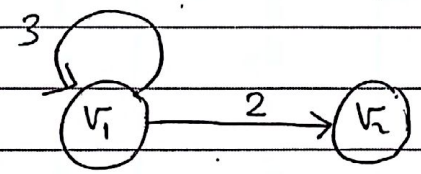
① تعتمد فكرته على وجود طريق مباشر فإذا كان هناك طريق مباشر مع الإنتباه إلى الإتجاه نضع القيمة



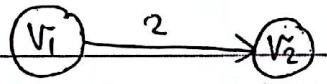
from V_1 to V_2 = $\boxed{2}$ (✓)

from V_1 to V_2 (X) = $\boxed{\infty}$

② في حال عدم وجود edge يشير إلى نفس الـ node نضع (صفر)

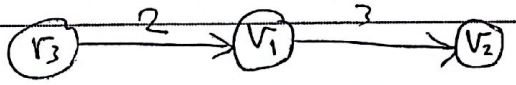


from V_1 to V_1 = 3



from V_1 to V_1 = $\boxed{\infty}$

③ في حال عدم وجود طريق مباشر بين 2 node نضع صفر



from V_3 to V_2 = $\boxed{\infty}$

Scanned by CamScanner

2 طريقة الـ D :-

"shortest path"

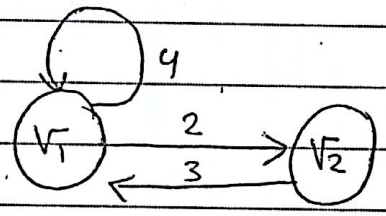
الطريق الأقصر.

① بغض النظر إذا كان هناك طريق مباشر بين 2 node

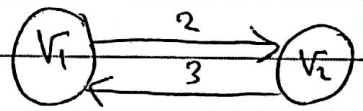
أولا ، نحن ننظر إلى أحقر جواب أو حل يوصلنا إلى النتيجة.

② في حال الـ node تشير إلى نفسها نضع الـ value

وإذا كانت لا تشير نضع $\boxed{0}$



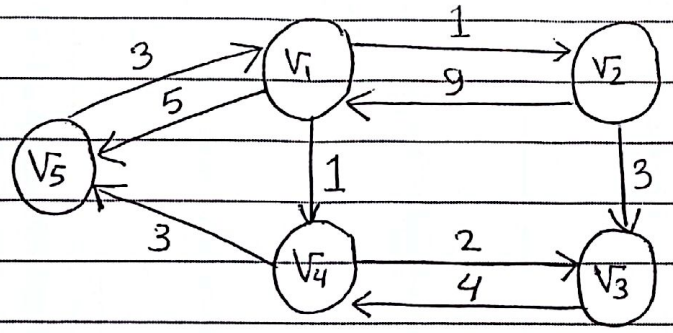
from V_1 to V_1 = $\boxed{4}$



from V_1 to V_1 = $\boxed{0}$

Scanned by CamScanner

example :-



Solution :-

	1	2	3	4	5
1	0	1	∞	1	5
2	9	0	3	2	∞
3	∞	∞	0	4	∞
4	∞	∞	2	0	3
5	3	∞	∞	∞	0

" W "

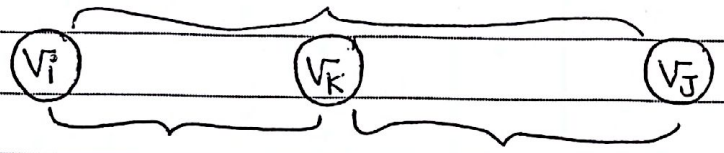
	1	2	3	4	5
1	0	1	3	1	4
2	8	0	3	2	5
3	10	11	0	4	7
4	6	7	2	0	3
5	3	4	6	4	0

" D "

Scanned by CamScanner

The shortest path use V_K

A shortest path from V_i to V_j using only vertices in $\{V_1, V_2, \dots, V_K\}$



A shortest path from V_i to V_K using only vertices in $\{V_1, V_2, \dots, V_K\}$

A shortest path from V_K to V_j using only vertices in $\{V_1, V_2, \dots, V_K\}$

The rule :-

$$D^{(K)}[i][j] = \text{minimum}(D^{(K-1)}[i][j], D^{(K-1)}[i][K] + D^{(K-1)}[K][j])$$

example 1 :-

$$D^{(3)}[2][4] = \text{minimum}(D^{(2)}[2][4], D^{(2)}[2][3] + D^{(2)}[3][4])$$

example 2 :-

$$D^{(4)}[7][4] = \text{minimum}(D^{(3)}[7][4], D^{(3)}[7][4] + D^{(3)}[4][4])$$

example 3 :-

$$D^{(5)}[2][1] = \text{minimum}(D^{(4)}[2][1], D^{(4)}[2][5] + D^{(4)}[5][1])$$

Scanned by CamScanner

The average search Time

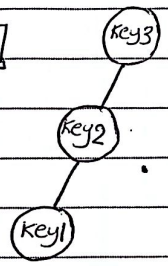
example :-

$P_1 = 0.7$

$P_2 = 0.2$

$P_3 = 0.1$

1



$3 \times 0.7 + 2 \times 0.2 + 1 \times 0.1 = \boxed{2.6}$

2

$3 \times 0.7 + 1 \times 0.2 + 2 \times 0.1 = \boxed{2.5}$

The End of Chapter 3 ^ ^

Scanned by CamScanner

Chapter 6 :-

B.2 / Solving Recurrences Using the Characteristic Equation :-

B.2.1 / Homogeneous Linear Recurrences :

example 1 :-

suppose we have the recurrence

$$t_n - 5t_{n-1} + 6t_{n-2} = 0 \quad \text{for } n > 1$$

$$t_0 = 0$$

$$t_1 = 1$$

Solution :-

$\boxed{t_n = r^n}$

$$r^n - 5r^{n-1} + 6r^{n-2} = 0$$

$$r^2 - 5r + 6 = 0$$

$$(r-3)(r-2)$$

$$\boxed{r=3} \quad \boxed{r=2}$$

$$t_n = C_1 3^n + C_2 2^n$$

$$t_0 = C_1(3)^0 + C_2(2)^0 = 0 \Rightarrow C_1 + C_2 = 0$$

$$t_1 = C_1(3)^1 + C_2(2)^1 = 1 \Rightarrow 3C_1 + 2C_2 = 1$$

تكملة الحل

Scanned by CamScanner

$$3C_1 + 2C_2 = 1$$
$$C_1 + 2C_2 = 0 \quad \times 3$$

$$\Rightarrow 3C_1 + 2C_2 = 1$$
$$3C_1 + 3C_2 = 0 \quad (-)$$

$$0 - C_2 = 1 \Rightarrow \boxed{C_2 = -1}$$

$$C_1 - 1 = 0 \Rightarrow \boxed{C_1 = 1}$$

The final answer:-

$$t_n = (3)^n - (2)^n$$

بتعويض قيم C_1 و C_2

$\Rightarrow$

Scanned by CamScanner

example 2 :-

suppose that :→ $t_n - 3t_{n-1} - 4t_{n-2} = 0$, for $n > 1$

$t_0 = 0$

$t_1 = 1$

Solution :-

$\boxed{t_n = r^n}$

$r^n - 3r^{n-1} - 4r^{n-2} = 0$

$r^2 - 3r - 4 = 0$

$(r-4)(r+1)$

$r = 4 \quad r = -1$

$t_n = C_1(4)^n + C_2(-1)^n$; $t_0 = C_1(4)^0 + C_2(-1)^0 = 0$

$\boxed{C_1 + -C_2 = 0}$

$t_1 = C_1(4)^1 + C_2(-1)^1 = 1$

$\boxed{4C_1 - C_2 = 1}$

$$\begin{array}{r} 4C_1 - C_2 = 1 \\ + \quad C_1 + C_2 = 0 \\ \hline 5C_1 + 0 = 1 \end{array}$$

$\boxed{C_1 = \frac{1}{5}}$

$\boxed{C_2 = -\frac{1}{5}}$

The final answer is :-

$t_n = \frac{1}{5}(4)^n - \frac{1}{5}(-1)^n$

Scanned by CamScanner

example 3 :-

suppose that : $t_n - 7t_{n-1} + 15t_{n-2} - 9t_{n-3} = 0$, for $n > 2$

$t_0 = 0$

$t_1 = 1$, $t_2 = 2$

Solution :-

$\boxed{t_n = r^n}$

$r^n - 7r^{n-1} + 15r^{n-2} - 9r^{n-3} = 0$

$r^3 - 7r^2 + 15r - 9 = 0$

$(r-1)(r-3)^2$

$\boxed{r=1}$ $\boxed{r=3}$ $\boxed{r=3}$

$t_n = C_1(1)^n + C_2(3)^n + nC_3(3)^n$

$t_0 = C_1(1)^0 + C_2(3)^0 + \cancel{(0)C_3(3)^0}^{\,0} \Rightarrow \boxed{C_1 + C_2 = 0}$

$t_1 = C_1(1)^1 + C_2(3)^1 + (1)C_3(3)^1 = 1 \Rightarrow \boxed{C_1 + 3C_2 + 3C_3 = 1}$

$t_2 = C_1(1)^2 + C_2(3)^2 + (2)C_3(3)^2 = 2 \Rightarrow \boxed{C_1 + 9C_2 + 18C_3 = 2}$

solving this system yields :-

$C_1 = -1$, $C_2 = 1$, $C_3 = \frac{1}{3}$

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

Scanned by CamScanner

6.2.5 / Nonhomogeneous Linear Recurrences.

example :-

suppose that :

$t_n - 3t_{n-1} = 4^n$

$t_0 = 0$

$t_1 = 4$

rule

for example

$b \to 4^n(2n^1 + 1)$

d

$(r-b)^{d+1}$

Solution ⇒

في الحل نقسم السؤال إلى قسمين ما قبل المساوي وما بعد المساوي

① $t_n - 3t_{n-1} = 0$

$r^n - 3r^{n-1} = 0$

$r - 3 = 0$

$\boxed{r = 3}$

② $b \to 4^n = 0$

على القانون نطبق $(r-b)^{d+1}$

$(r-4)$

$\boxed{r = 4}$

$t_n = C_1(3)^n + C_2(4)^n$

$t_0 = C_1(3)^0 + C_2(4)^0 = 0$

$\boxed{C_1 + C_2 = 0}$

$t_1 = C_1(3)^1 + C_2(4)^1 = 4$

$\boxed{3C_1 + 4C_2 = 4}$

$3C_1 + 4C_2 = 4$

$C_1 + C_2 = 0 \quad \boxed{\times 3}$

$\boxed{-}$ $3C_1 + 4C_2 = 4$

$3C_1 + 3C_2 = 0$

$\boxed{C_2 = 4}$

$\boxed{C_1 = -4}$

$t_n = -4(3)^n + 4(4)^n$

Scanned by CamScanner

example 9 :-

Suppose that :-

$t_n - 3t_{n-1} = 4^n(2n+1)$, for $n > 1$

$t_0 = 0$

$t_1 = 12$

Solution →

① $t_n - 3t_{n-1} = 0$

$r^n - 3r^{n-1} = 0$

$r - 3 = 0$

$\boxed{r = 3}$

② $\overset{b}{4}^n(2n^{\overset{d}{1}} + 1) = 0$

on the rule :-

$(r-b)^{d+1} = (r-4)^2$

$\boxed{r = 4}$ $\boxed{r = 4}$

$t_n = C_1(3)^n + C_2(4)^n + nC_3(4)^n$

$t_0 = C_1(3)^0 + C_2(4)^0 + \cancel{(0)C_3(4)^0}$

$= \boxed{C_1 + C_2 = 0}$

$t_1 = C_1(3)^1 + C_2(4)^1 + (1)C_3(4)^1 = 12$

$= \boxed{3C_1 + 4C_2 + 4C_3 = 12}$

* we have three vairables , so we should know the value for t_2 to know the third equation.

$t_2 - 3t_1 = 4^2(2 \times 2 + 1)$ and $t_1 = 12$

$t_2 = 3 \times 12 + 80 = 116$;

$t_2 = C_1(3)^2 + C_2(4)^2 + (2)C_3(4)^2 = 116$ ⟹ 3 vales

Scanned by CamScanner

$C_1 + C_2 = 0 \quad \ldots (1)$

$3C_1 + 4C_2 + 4C_3 = 12 \quad \ldots (2)$

$9C_1 + 16C_2 + 32C_3 = 116 \quad \ldots (3)$

$C_1 = 20 \quad , \quad C_2 = -20 \quad , \quad C_3 = 8$

$t_n = 20(3)^n - 20(4)^n + 8n(4)^n$

Scanned by CamScanner

B.3/ solving Recurrences by substitution

example 1 :-

$$t_n = t_{n-1} + n \quad , \text{for } n > 1$$
$$t_1 = 1$$

Solution :-

$$t_n = t_{n-1} + n$$
$$t_{n-1} = t_{n-2} + n - 1$$
$$t_{n-2} = t_{n-3} + n - 2$$
$$\vdots$$
$$t_2 = t_1 + 2$$
$$t_1 = 1$$

now by substitution

$$t_n = t_{n-1} + n$$
$$= t_{n-2} + n - 1 + n$$
$$= t_{n-3} + n - 2 + n - 1 + n$$
$$\vdots$$
$$= t_1 + 2 + \dots + n - 2 + n - 1 + n$$
$$= 1 + 2 + \dots + n - 2 + n - 1 + n$$

Scanned by CamScanner

example 2 :-

$$t_n = t_{n-1} + \frac{2}{n} \quad , \quad n > 1$$
$$t_1 = 0$$

Solution ⇒

$$t_n = t_{n-1} + \frac{2}{n}$$
$$t_{n-1} = t_{n-2} + \frac{2}{n-1}$$
$$t_{n-2} = t_{n-3} + \frac{2}{n-2}$$
$$\vdots$$
$$t_2 = t_1 + \frac{2}{2}$$
$$t_1 = 0$$

now by substitution :-

$$t_n = t_{n-1} + \frac{2}{n}$$
$$= t_{n-2} + \frac{2}{n-1} + \frac{2}{n}$$
$$= t_{n-3} + \frac{2}{n-2} + \frac{2}{n-1} + \frac{2}{n}$$
$$\vdots$$
$$= t_1 + \frac{2}{2} + \dots + \frac{2}{n-2} + \frac{2}{n-1} + \frac{2}{n}$$
$$= 0 + \frac{2}{2} + \dots + \frac{2}{n-2} + \frac{2}{n-1} + \frac{2}{n}$$

Scanned by CamScanner

Theorem 6.5

suppose a complexity function $T(n)$ is eventually nondecreasing and satisfies.

$$\boxed{\begin{array}{l} T(n) = aT\left(\frac{n}{b}\right) + cn^k, \quad \text{for } n > 1, n \text{ a power of } b \\ T(1) = d \end{array}}$$

wher $b \geq 2$ and $k \geq 0$ are constant integers, and a, c and d are constants such that $a > 0$, $c > 0$, and $d \geq 0$ Then:-

$$T(n) \in \begin{cases} \Theta(n^k) & \text{if } a < b^k \\ \Theta(n^k \lg n) & \text{if } a = b^k \\ \Theta(n^{\log_b a}) & \text{if } a > b^k \end{cases}$$

$\Leftarrow$ "lip"

Scanned by CamScanner

example 1 :-

$$T(n) = \overset{a}{8}\, T(n/\overset{b}{4}) + 5 n^{\overset{k}{2}}$$

$$T(1) = 3$$

$$8 \qquad 4^2$$

$$8 < 16$$

$$\therefore T(n) \in \Theta(n^2)$$

example 2 :-

$$T(n) = \overset{a}{9}\, T(n/\overset{b}{3}) + 5 n^{\overset{k}{1}}$$

$$T(1) = 7$$

$$9 \qquad 3^1$$

$$9 > 3$$

$$\therefore T(n) \in \Theta(n^{\log_3 9})$$

The End of chapter 6 ^_^

Scanned by CamScanner

Chapter 4 :- Greedy Approach

Using :-
used to solve optimization problems.

⇓

It has more than one solution.

⇒ Basic components in greedy approach?

1] selection procedure :- choose the next item to add to the set.
The selection is preformed according to a greedy criterion that satisfies some locally optimal consideration at the time.

2] feasibility check :- Determine if the new set is feasible by checking whether it is possible to complete this set in such a way as to give a solution to the instance.

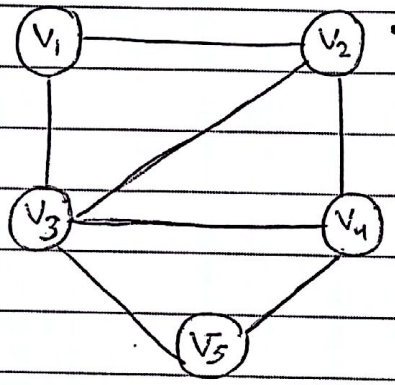
3] Solution check :- determines whether the new set constitutes a solution to the instance.

Scanned by CamScanner

Minimum Spaning tree :-

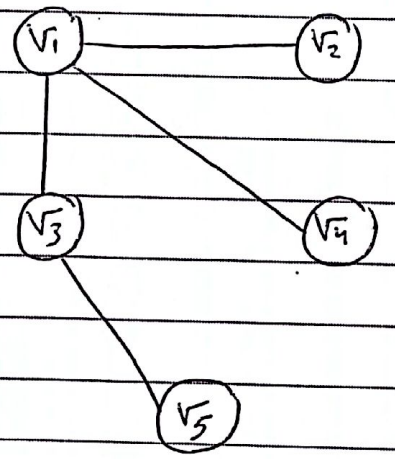
⊙ في البداية يجب أن نعلم أن "tree" هو شكل من أشكال الـ "graph".

⊙ ولكن الإختلاف بينهم سيوضح عن طريق هذا المثال :-



" graph "
↓
⊙ يوجد Loop .

1) Undirected graph
2) connected graph



" tree "
↓
⊙ لا يوجد Loop .

Scanned by CamScanner

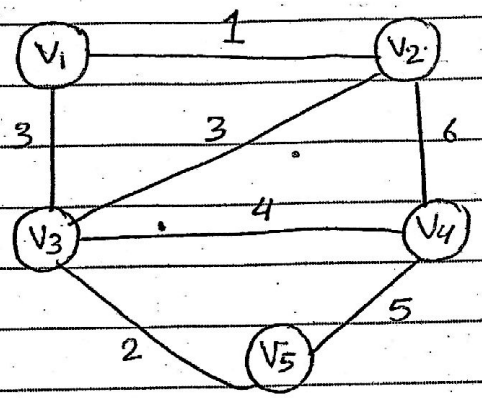
we can find minimum spaning tree by 3 algorithems :-

1. prim's algorithem.
2. Kruskal's algorithem.
3. Dijkstra's algorithem.

يتبع

Scanned by CamScanner

[1] prims Algorithem :-

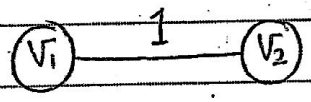


① هو الذي يحدد أي vertex يجب أن تختار وبالعادة يكون vertex 1 .

① select vertex 1 :-

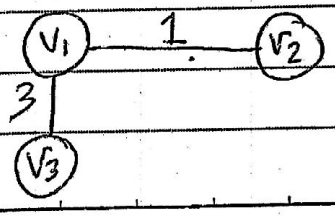
② نلاحظ أنّ V_1 مرتبطة مع V_2 و V_3 تختار الطريق الأقرب لـ V_1 وهو V_2 :-

② select V_2 :-



③ نلاحظ أن V_1 مرتبطة مع V_3 بـ (3) و V_2 مرتبطة مع V_3 بـ (3) ومع V_4 بـ (6)
نختار الطريق الأقصر بالنسبة لكليهما .

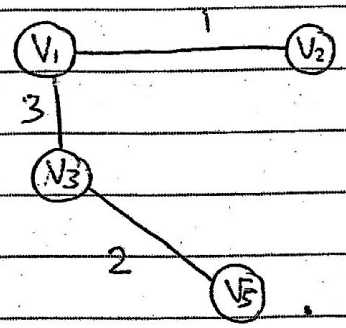
③ select V_3 :-



Scanned by CamScanner

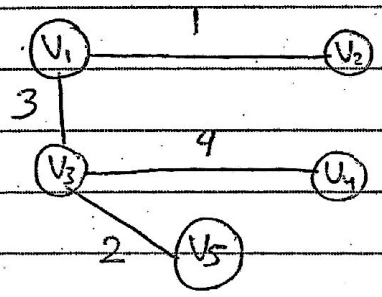
4 نختار الطريق الأقصر بالنسبة لـ V_1 و V_3 و V_3 :-

(4) select V_5 :-



5 دفعي لنا V_4 نختار الطريق الأقصر الذي يؤدي لها :-

(5) select V_4 :-



ملاحظة :-

الهدف هو أن نصل إلى جميع الـ vertex ولكن يجب أن يكون الطريق الأقصر الذي يؤدي إليها وتذكر لا يجوز أن يكون هناك loop.

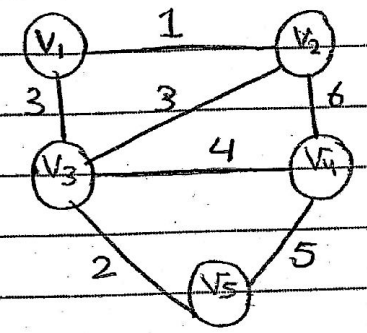
Time complexity :-

$$T(n) \in \Theta(n^2)$$

Scanned by CamScanner

[2] Kruskal's algorithem

↳ edges are sorted by weight.



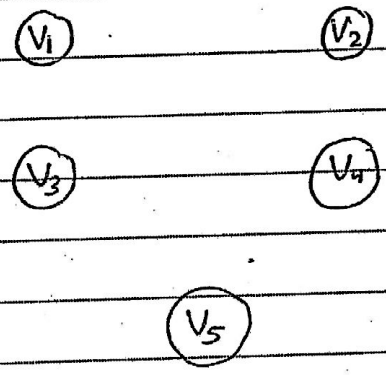
[1] firstly we will sort the edges by weight.

#سوف نرتبهم من الأقل إلى الأعلى.

(V_1, V_2)	1
(V_3, V_5)	2
(V_1, V_3)	3
(V_2, V_3)	3
(V_3, V_4)	4
(V_4, V_5)	5
(V_2, V_4)	6

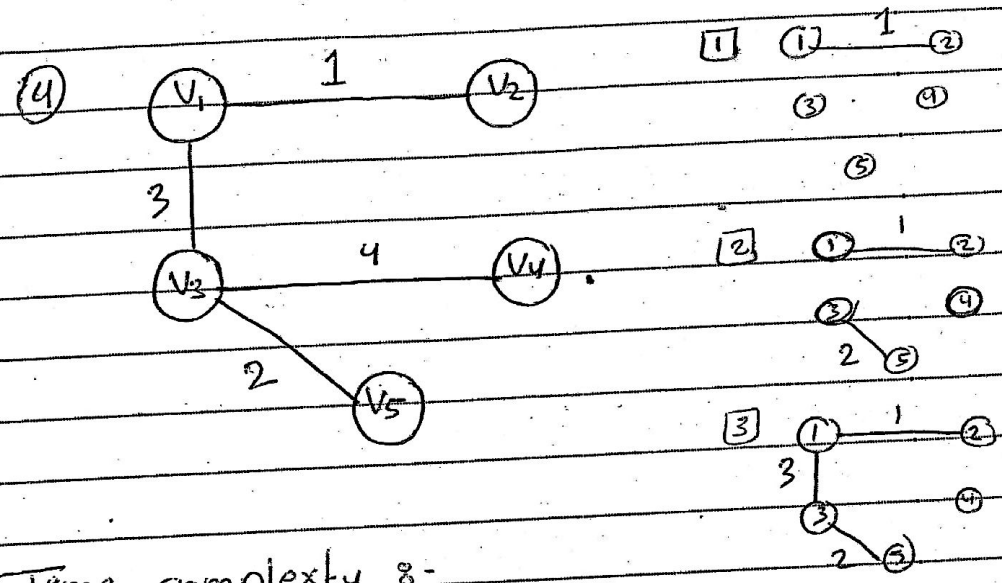
تستخدم واحدة فقط حتى لا يقل عليه

[2] we will put the nodes "vertex" without any connecting by edges.



Scanned by CamScanner

[3] select the edges between two vertex wich have the minimum value.



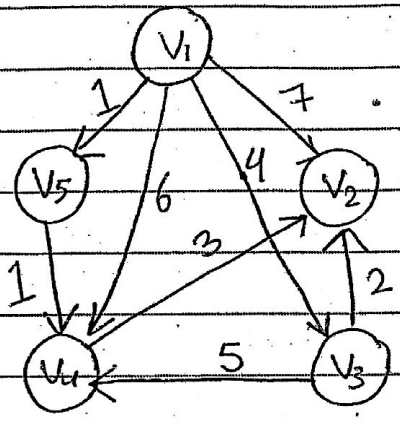
Time complexty :-

$$W(n) \in (n^2 \lg n)$$

Scanned by CamScanner

3 Dijkstra's algorithm for single source shortest path.

أقصر طريق من node معينة.



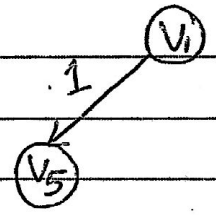
الفكرة هو shortest path من node معينة في السؤال هو الذي نقوم بتحديدها.

for example in this question :-

1 compute shortest path from V_1.

بالنسبة لـ V_1 ⇐ V_1

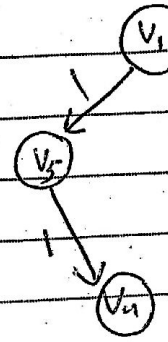
2 أقرب node لـ V_1 هي V_5



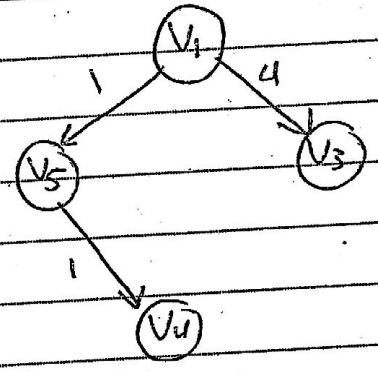
Scanned by CamScanner

3 بالنسبة لـ V_1 أقرب طريق بعد V_5 هو V_4 :-

* ملاحظة يمكن استخدام vertex وسيط :-

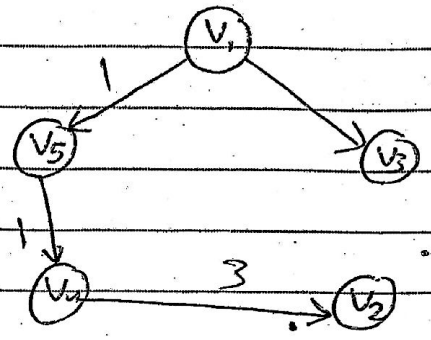


4 بالنسبة لـ V_1 أقرب طريق من بعد V_5 و V_4 هو $\boxed{V_3}$



Scanned by CamScanner

5 أقرب طريق بالنسبة لـ V_1 لآخر node هو



* تم الحصول على shortest path عن طريق node معينة.

The End of chapter 4 ^_^

Scanned by CamScanner

Chapter 5 :- Backtracking

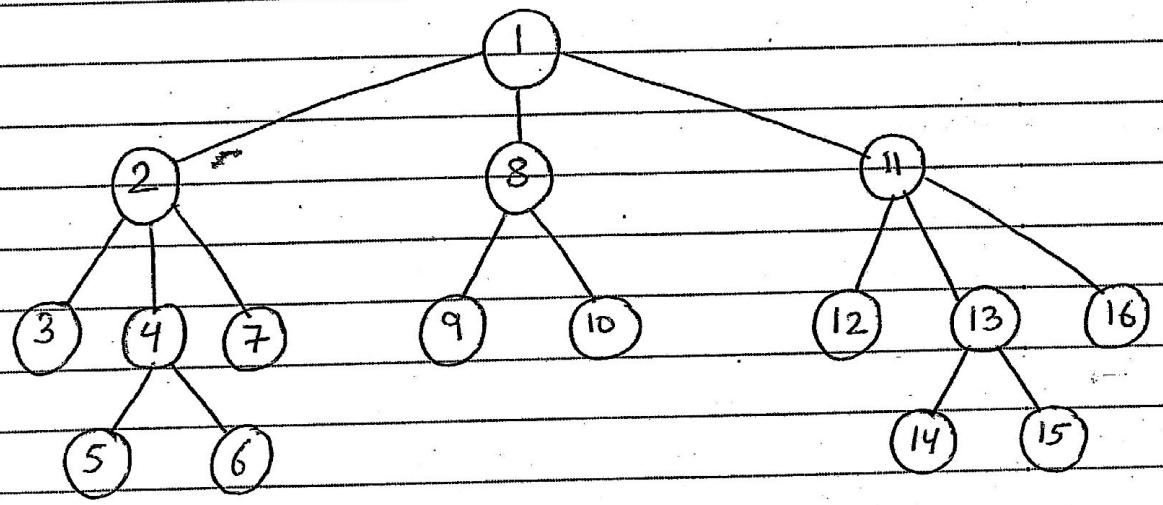
Backtracking :- It is away to solve some problems.

فكرة الـ Backtracking :-

من اسمها تراجع ، بعض الـ algorithims الأخرى تمر في جميع الحلول حتى وإن كانت خاطئة إلى أن تصل إلى الجواب النهائي الصحيح .
ولكن في الـ Backtracking إذا كان الحل من بدايته خاطئ فهي لا تعرفه .

firstly we will study "Depth first search" and by it we will understand Backtracking.

Depth first search :-



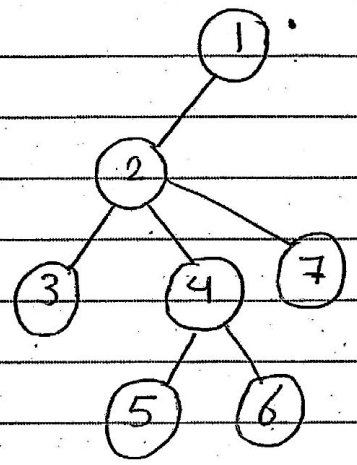
شرح ←

* for example, you have from 1 to 16 and you want to organize them in a tree :-

The rule is :-

root ⇒ Left ⇒ root

we called this "preorder tree"



⊙ we will use this rule "pre order tree" indepth first rule to solve n-Queens problem.

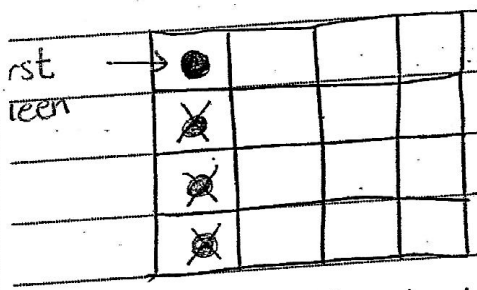
⇒

Scanned by CamScanner

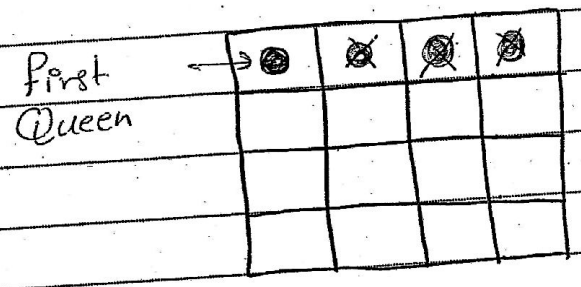
n Queens problem :- → we will use depth first search

راح نتعامل مع 4 Queens problem

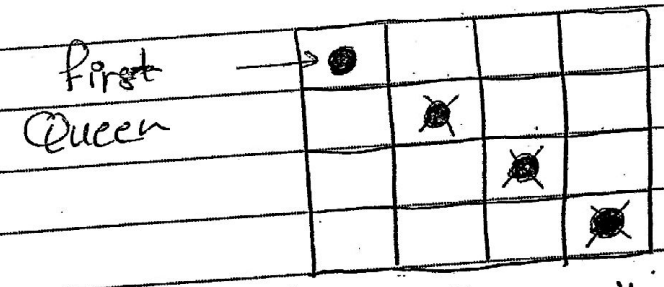
الفكرة وما فيها عندما نضع Queen معين في شطرنج وأردنا وضع الآخر لا يجوز أن يكون ① عمودي أو ② أفقي أو ③ قطري مع الـ Queen الأول.



لا يجوز وضع الـ Queen الثاني بصورة عمودية بالنسبة لـ Queen الأول

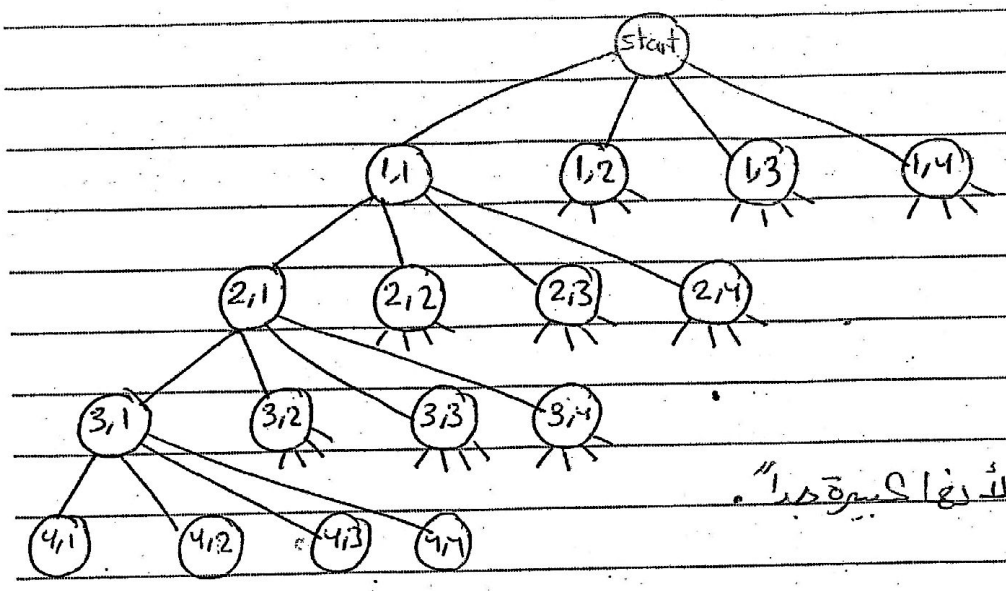


لا يجوز وضع الـ Queen الثاني بصورة أفقية بالنسبة لـ Queen الأول



لا يجوز وضع الـ Queen الثاني بصورة قطرية بالنسبة لـ Queen الأول.

Scanned by CamScanner



* الشجرة غير كاملة لأنها كبيرة جداً.

* عند تطبيق depth first search على 4_Queens problem لقد قمنا بالمرور على جميع العناصر وهذا ما يختلف عن Backtracking

* في n_Queens problem قمنا بالمرور على جميع العناصر ولكن الآن سوف نرى ماسوف يقوم به الـ backtracking.

State Space Tree :-

→ we will use backtracking

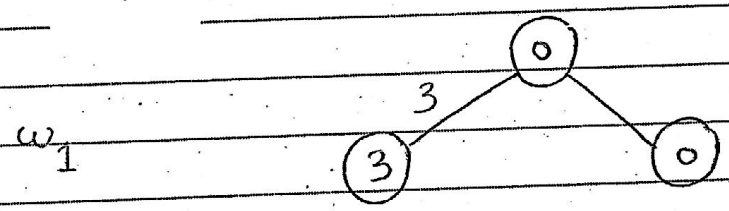
سوف تلاحظ أنه إذا كان الخطأ من بداية لن ندخل فيها أي أننا لن نصل على جميع العناصر.

هي عبارة عن "Binary tree".

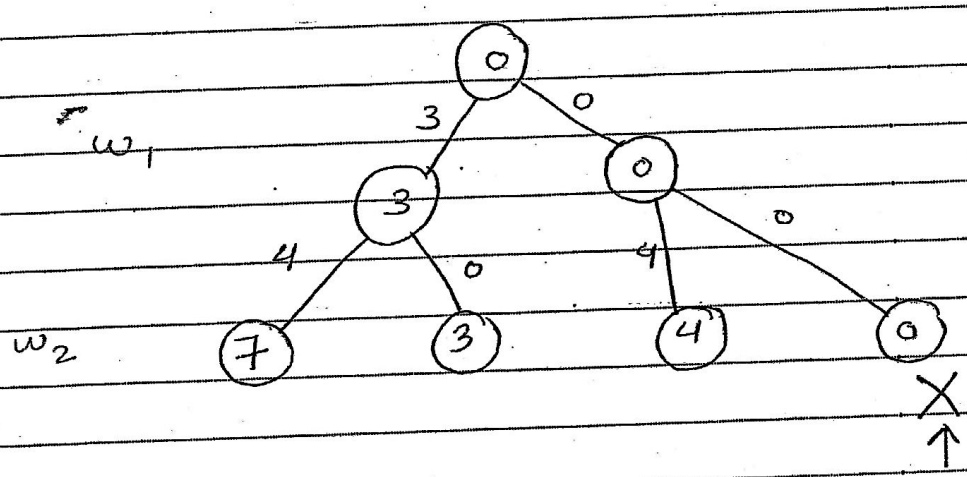
state space tree :-

example :- When $W = 13$ and $w_1 = 3, w_2 = 4, w_3 = 5, w_4 = 6$

$\boxed{1}$

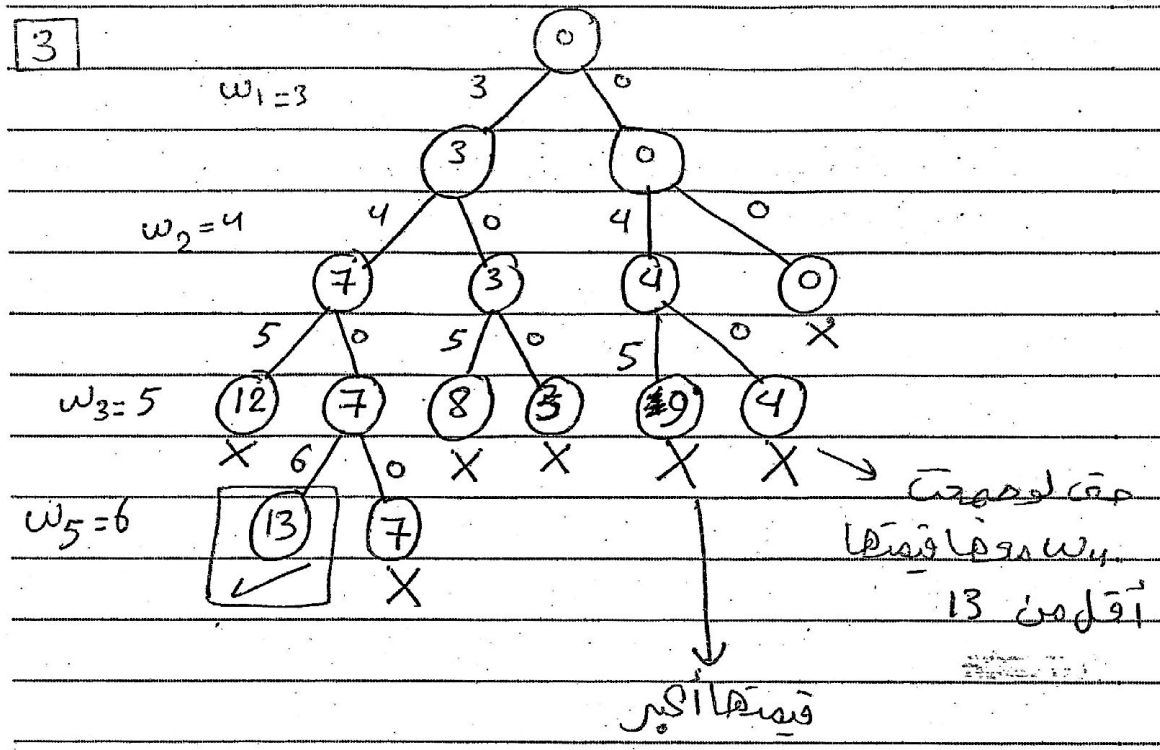


$\boxed{2}$



يتوقف الحل هنا والسبب
حتى لو جينا w_3 و w_4 حاصل
جمعهم أقل من $\boxed{13}$.

3



تم الوصول إلى الحل الصحيح بدون المرور في جميع العناصر.

Graph Coloring

The m-coloring problem :-

- Finding all ways to color an undirected graph using at most "m" different colors, so that no two adjacent vertices are the same color.

- Usually the m-coloring problem consider as a unique problem for each value of "m".

الفكرة هي :-

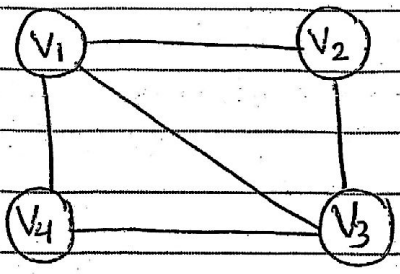
- لا يوجد 2 node تصل بينهم edge لهم نفس اللون

- في حال عدم توفر الألوان الكافية لـ problem يصبح هناك "no solution".

التوضيح ⟹
example.

Scanned by CamScanner

example :-



1 2-coloring problem :-

" no solution"

2 3-coloring problem :-

V_1	green
V_2	red
V_3	black
V_4	red

* كل 2 vertex بينهم edge مختلفين بالنسبة للون.

Scanned by CamScanner

إذا نظرنا إلى ال
بالنسبة للود

lane in such
each other.

corresponds

ple ⟹

→ Application: Coloring of maps

خرائط سنجد أن كل دولتين متقاربتين مختلفتين
فقط نستطيع تمييز دولة عن أخرى .

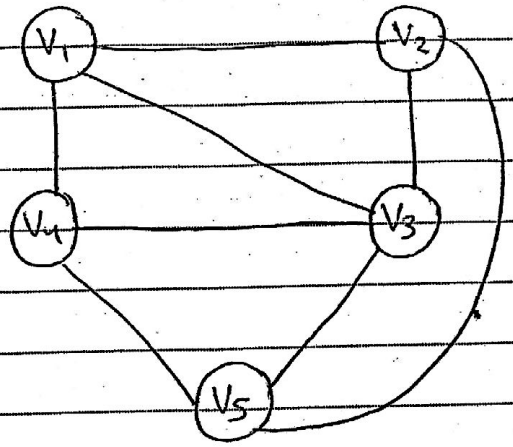
planar graph:-

- It can be drawn in a p
away that no two edges cross

- To every map there c
aplanar graph

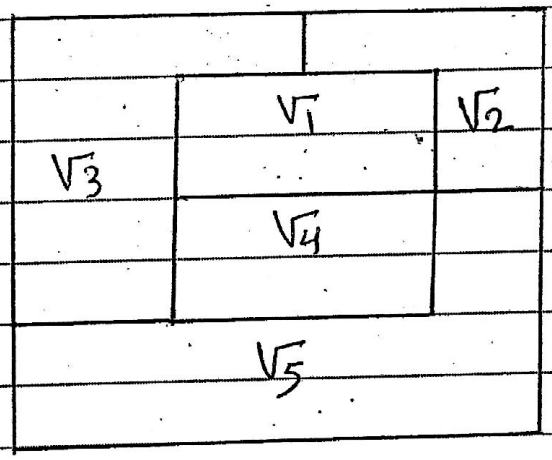
Exam

Scanned by CamScanner



هكذا يتم تمثيلها بال planar :-

corresponded planar graph :-



The End of Chapter 5

Scanned by CamScanner

Team ZerOne
Computer & Networks Engineers

ملخص

خوارزميات

عضو الجمعية الطلابية :

عبدالرحمن أبوزيد

قال عليه السلام :
{ من كان في حاجة أخيه كان الله في حاجته }

Scanned by CamScanner