Eng-Raed

تراكيب البيانات

في نهاية هذا الفصل سوف تتعلم :

- **مقدمة إلى هياكل البيانات**
- **أنواع هياكل البيانات.**
- **التعرف على المكدس , الطابور , القوائم , الأشجار.**
- **تمثيل البيانات الديناميكية.**

اعداد: أ. م. رائد خضير

2021 - 2020

1.1 مقدمة INTRODUCTION

من المألوف قبل الشروع إلى فهم شيء يجب معرفة توابعه وتفاعل هذه العناصر مع بيئتها المتوافرة فيها والعمليات التي يمكن أن تحدث على هذه العناصر التي تؤلف فيما بينها وحدة متناغمة مترابطة لكي توصل الفكرة إلى العقل بشكل جيد وسريع.
ومن الأشياء التي يجب تحديدها كمتخصصين في علم الحاسب الآلي هي البيانات DATA و المعلومات INFORMATION لإنها هي أساس تعاملنا مع الحاسب.
فما هي البيانات وما هي المعلومات وكيف يمكن لنا أن نحددها ؟
فالبيانات /هي مجموعة من الحقائق أو الأفكار قد تكون حروف أو أرقام أو صوراً أو خليطاً مما سبق.
والمعلومات /مجموعة من الحقائق والأفكار عن شيء ما تمت معالجتها .
إذن فالبيانات هي المادة الخام للمعلومات أو أن المعلومات هي البيانات بعد معالجتها ومن الواضح آن الفرق الأساسي بينهم هي المعالجة .

- أنواع المعالجة على البيانات

1. الإضافة.
2. الحذف أو الإلغاء .
3. الدمج .
4. الفرز .
5. التحليل والتركيب باستخدام العمليات الحسابية (+ ,*,- ,/)والعمليات المنطقية (= ,=!, =>, <= ,>, <) .
6. النسخ الالكتروني (SAVE).
7. الحماية والفك .
8. الاسترجاع والتعديل والتخزين .

إذن لكي نحصل على معلومات لابد من الحصول على بيانات أولاً ثم القيام بمعالجة هذه البيانات معالجة صحيحة.

- التركيب الفيزيائي والتركيب المنطقي للبيانات

نجد أن سرعة معالجة البيانات تعتمد كثيراً على عدة عوامل أضافه إلى العامل الزمني اللازم للمعالجة ومن أهميتها

- **ü** عوامل تحدد من الذاكرة الرئيسية.
- **ü** عوامل تحدد من وحدات الإدخال والإخراج.
- **ü** عوامل تحدد من تفاعل الإدخال والإخراج مع الذاكرة الرئيسية(تبادل المعلومات بين هذه الوحدات أي عملي مقايضة.

نجد أن العامل الأول يتطلب وجود برنامج للمعالجة والنسبة للعامل التأني فأنة يتطلب سرعة للوصول إلى المعلومات لإحضارها من الوحدات الإدخال , والذي يهمنا هو الإقلال من عملية المقايضة بين الذاكرة الرئيسية ووحدات الإدخال والإخراج ولهذا لابد من الإشارة إلى التركيب الفيزيائي والمنطقي للبيانات حيت نجد تعريف الاثنين باختصار
التركيب المنطقي / هو وجه نظر المبرمج في سير البرنامج أي آن هذا ترتيب معلومات البرنامج بشكل معين حتى يتم تنفيذ هذا البرنامج بطريقة صحيحة .
التركيب الفيزيائي / وهو يعني كيفية ترتيب البيانات على أوساط التخزين مثل الشريط المغناطيسي والقرص المغناطيسي حيت تخزن البيانات على القرص المغناطيسي بطريقة مباشرة أو تتابعيه أي تسلسلية مفهرسة.

هياكل البيانات وتراكيب البيانات وبني المعطيات لهما نفس المعنى /عبارة عن آليات وخوارزميات معينة توضع لبرامج بحيث تطبق بشكل جيد فهي مفيد جدا في برمجه قواعد البيانات و تساعد على تنفيذ مهام وتسهيل مهام من مهام الكمبيوتر ومن استغلال مواقع الذاكرة بشكل جيد ومنظم ويجب على المبرمج تطيق هذه الآليات بشكل جيد وإلا خلت من معنى الخوارزمية .
وبتعريف منطقي له /هي طريقه ترابط و ترص البيانات مع بعضها البعض في الذاكرة بحيث هذه البيانات تتخذ شكلا وهيكلاً معيناً في الذاكرة فتعتبر بنية عضوية لمجموعة من عناصر البيانات المتطابقة نوعاً
وشكلاً والتي تنظم في نسق واحد لتؤدي غرضاً محدداً.

1.2 فوائد هياكل البيانات

- التحكم في توزيع البيانات و التعرف إلى طبيعتها وبنائها الأساسي بنسق معين في الذاكرة.
- بناء برامج قوية ومتماسكة من حيت البناء والمنطق.
- تمكين المبرمج من أبداع طرق مبتكرة في كتابة البرامج المختلفة.
- اختصار زمن التخزين واسترجاع البيانات من الذاكرة .

1.3 أنواع هياكل البيانات

- هياكل بيانات ثابتة ساكنة (STATIC INFORMATION).

كالمتجهات والجداول والسجلات وعند التصريح عنها فيجب تحدي حجم هذه البيانات فلا تقبل الإضافة فوق حجمها المحدد .....

- هياكل بيانات شبكية.
- هياكل بيانات ديناميكية إي متحركة متغيرة

وينقسم هذا النوع إلى نوعين

1)هياكل بيانات خطية متغيرة / وهي التي تنظم في خط متتالي

- الملفات .
- القوائم .
- الطوابير .
- المكدسات .
- الأبجديات .
- المجموعات .

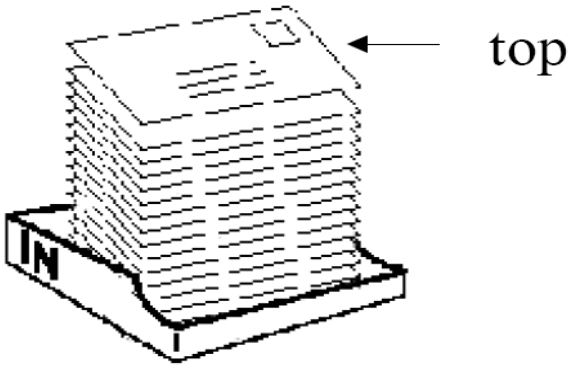
2)هياكل بيانات متشعبة إي بشكل عشوائي مخزنة في الذاكرة / مثل الأشجار , الخرائط

وسنتكلم عن هذه المواضيع مبدئياً بالهياكل الاستاتيكية بواسطة المتجهات .....

1.4 المكدس (Stack)

وهو عبارة عن نموذج خاص لتخزين البيانات بالية ثابتة وإخراجها باليه ثابتة بشكل مؤقت وهو عبارة عن صندوق توضع به البيانات بالية الداخل أولا الخارج أخرا والداخل أخرا الخارج أولا LIFO (LAST INPUT FIRST OUTPUT)

وكمثال بسيط أيضا نشبه عملة بقشطه المسدس الرصاصة الأولى تخرج أخر شيء والرصاصة الأخيرة تخرج أولاً ولهذا فان الإضافة تتم من الأعلى والحذف و يوجد مؤشر واحد يسمى top .
والقراءة أيضا يتم من الأعلى إي من طرف واحد عن طريق top كما في الشكل 1-15.



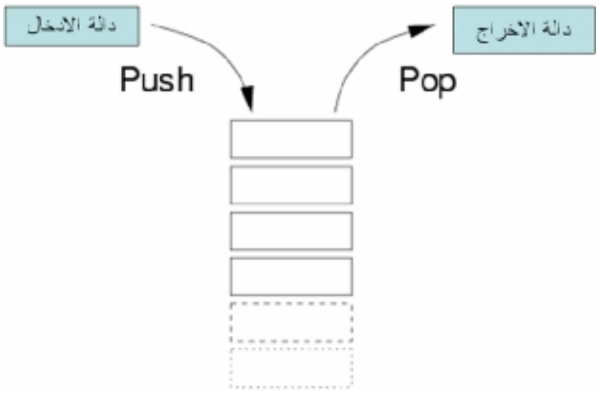
شكل 1-15

1.4.1 فوائد المكدس

- **ü** إيجاد قيم التعابير الحسابية
- **ü** يستخدم لغايات الاستدعاء الذاتي
- **ü** في عمليات الاعتراض والمقاطعة المستخدمة بالويندوز
- **ü** استدعاء البرامج الفرعية

ومن الإشكال السابقة نجد أن المكدس لا يحتوي إلا على مؤشر واحد فقط TOP

فعندما يكون المكدس فارغاً فأن TOP=-1 وعند إدخال أول قيمة فإننا نزيد من قيمة TOP ++ وكل ما أدخلنا قيمة فان المؤشر يزيد بمقدار واحد إلى أن يمتلئ المكدس والشكل 2-15 يبين ذلك.



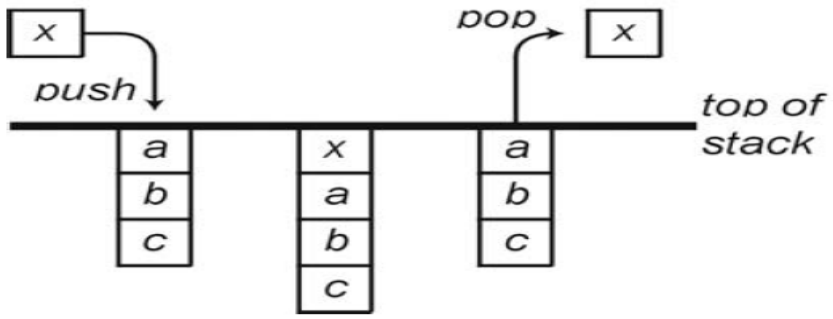
شكل 2-15

وعملية أخراج القيم من المكدس فإننا ننقص المؤشر بمقدار واحد أيضا إلى أن يصل قيمة المؤشر = -1 أو NULL فبهاكدا يكون المكدس فارغاً.

1.4.2 طرق تمثيل المكدس وتخزين عناصره في الذاكرة وتامين عملية بلوغها

- **ü** التمثيل المترابط الحلقي لعناصر المكدس على شكل لائحة إي على شكل قائمة .

ü التمثيل المتراص (COMPACT) للعناصر في الذاكرة إي على شكل مصفوفة أحادية .
وكمثال على تمثيل المكدس بالمتجهات لننظر إلى الشكل 3-15



شكل 3-15

ولنفترض أن لدينا مصفوفة حجم (4) وأدخلنا أخر قيمة (X) وإذا أردنا إخراج قيمة فان المكدس سيعطي لنا أخر قيمة دخلت وهي (x) كما في الشكل 3-15
وهذا أول برنامج له يعمل على إدخال قيم داخل المكدس ثم يقوم بطباعته

```
1. //ArrayStack
2. import java.io.*;
3. class Array_Stack1 {
4.   static final int CAPACITY = 5;
5.   static int[] Stack1 = new int[CAPACITY];
6.   static int top = -1;
7.
8.   static boolean isEmpty(){return (top < 0);}
9.
10.   static boolean isFull() {return (top+1== CAPACITY);}
11.
12.   static void push(int element){
13.     if (isFull())
14.                          System.out.println("Stack is full.");
15.                          else
16.                          Stack1[++top] = element;
17.                        }
18.
19.   static int pop() {
20.
21.     if (isEmpty()){
22.                           System.out.println("Stack          is
empty.");
23.                           System.exit( 0 );
24.                          }
25.     return Stack1[top--];
26.               }
```

```
27.
28. public static void main(String args[])throws IOException {
29.    String num;
30.    BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
31.    System.out.println( "Enter first integer" );
32.    while(!isFull())
33.                              {num=br.readLine();
34.                                   push(Integer.parseInt(num));
35.                              }
36.
37.    while(!isEmpty())System.out.println(pop()+" ");
38.
39. }
40. }
```

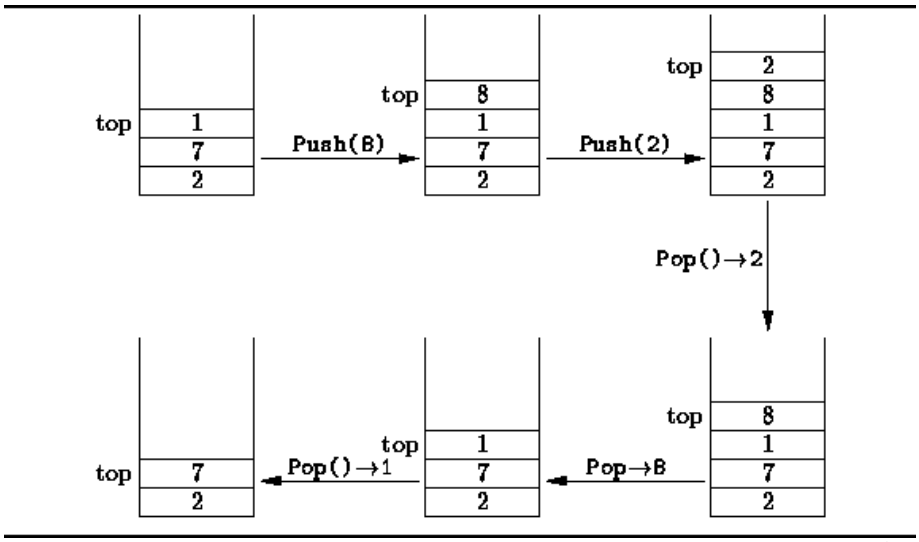
شرح المثال:
في السطر 6 تم تعريف Top ويسمى ذيل المكدس وهو متغير عام تستطيع الدوال الوصول إلية وتغير قيمته .
السطر 19 دالة أخراج البيانات من المكدس
السطر 12 دالة إدخال البيانات من المكدس وهي تأخذ بارا متر من نوع مصفوفة ومن نوع أنتجر

نعلم أن المكدس ممتلئ عندما يكون مؤشر الذيل أي top = حجم المصفوفة -1
وإلا سنزيد من المؤشر بواحد وسنضع القيمة بداخل المصفوفة بداخل الموقع الذي تكون قيمته top

STACK TOP ,PUSH,POP كل هذه التعابير عبارة عن أسماء متغيرات وليس من الضروري التقيد بهذه الأسماء فهي ليست دوال .

نعلم أن المكدس أصبح فارغاً عندما تكون قيمة المؤشر top =-1 أي اقل من الصفر
وإلا سنخرج القيمة من داخل المصفوفة التي تحمل عنوان قيمة top وسنطرح قيمة المؤشر بواحد .

فيكون ناتج تنفيذ البرنامج كالتالي:



شكل 4-15

ومن المثال السابق والشكل 4-15 يتبين لنا إليه عمل المكدس فيا أحبابي لا يخيفكم هذا المصطلح الغريب STACK مصفوفة ولا يختلف عنها إلى بشيء واحد ألا وهي إليه عملة التي ذكرناها سابقاً.

وكمثال أخر سنقوم باستخراج اكبر قيمة بالمكدس فقط سيكون التغير في دالة ألإخراج pop

```
1. //استخراج اكبر عدد من المكدس
2. import java.io.*;
3. class Array_Stack2 {
4.   static final int CAPACITY = 5;
5.   static int[] Stack1 = new int[CAPACITY];
6.   static int top = -1;
7.
8.   static boolean isEmpty(){return (top < 0);}
9.
10.   static boolean isFull() {return (top+1== CAPACITY);}
11.
12.   static void push(int element){
13.     if (isFull())
14.                          System.out.println("Stack is full.");
15.                          else
16.                          Stack1[++top] = element;
17.                        }
18.
19.   static int pop() {
20.
21.     if (isEmpty()){
22.                           System.out.println("Stack          is
empty.");
23.                           System.exit( 0 );
24.                          }
```

```
25.     return Stack1[top--];
26.                 }
27.   static int max(){ int temp=pop(),temp2;
28.                                 while(!isEmpty()){
29.                                                 temp2=pop();
30.
if(temp<temp2)temp=temp2;
31.                                                 }
32.                                  return temp;
33.                                 }
34.
35. public static void main(String args[])throws IOException {
36.    String num;
37.    BufferedReader    br    =    new    BufferedReader(new
InputStreamReader(System.in));
38.    System.out.println( "Enter first integer" );
39.    while(!isFull())
40.                             {num=br.readLine();
41.                                 push(Integer.parseInt(num));
42.                             }
43.
44.    System.out.println(max());
45.
46. }
47. }
```

شرح المثال:
اعتقد أن المثال واضح وبسيط ولا يوجد بة إي تعقيد فقط الاختلاف بينة وبين البرنامج السابق هو إضافة منهاج max التي تعيد لنا أكبر قيمة بداخل المكدس.

إلا الآن اعتقد قد تبثث فكرة المكدس والية عملة في ذهنك , الآن إذا طلب منك إن تدخل بيانات إلى المكدس وتعكس المكدس فكيف ذلك سيكون فكر قليلاً وتذكر إلية عمل المكدس ولا تقول نقوم بطباعة المصفوفة من البداية فهذا ليس صحيحاً فقد خليت من عملة .....
ها هل أتت الفكرة بعقلك حاول ولا تستعجل.....
يبدو لي أن الفكرة لم تأتي إليك إذن صلي على نبيك وتتبع البرنامج التالي بهدوء

```
1. // برنامج يعمل على عكس مكدس
2. import java.io.*;
3. class Stack_inv {
4.   static final int CAPACITY = 5;
5.   static int[] Stack1 = new int[CAPACITY];
6.   static int top = -1;
7.
8.   static boolean isEmpty(){return (top < 0);}
9.
10.   static boolean isFull() {return (top+1== CAPACITY);}
```

```
11.
12.  static void push(int element){
13.    if (isFull())
14.                          System.out.println("Stack is full.");
15.                          else
16.                          Stack1[++top] = element;
17.                        }
18.   static int pop() {
19.
20.    if (isEmpty()){
21.                           System.out.println("Stack          is
empty.");
22.                           System.exit( 0 );
23.                          }
24.    return Stack1[top--];
25.              }
26.  static void rev(){
27.                               int[]      Stack2      =      new
int[CAPACITY];
28.                               int[]      Stack3      =      new
int[CAPACITY];
29.                               int top2=-1,top3=-1;
30.
31.
       while(top>=0)Stack2[++top2]=Stack1[top--];
32.
       while(top2>=0)Stack3[++top3]=Stack2[top2--];
33.
       while(top3>=0)Stack1[++top]=Stack3[top3--];
34.                          }
35.
36. public static void main(String args[])throws IOException {
37.   String num;
38.   BufferedReader    br    =    new    BufferedReader(new
InputStreamReader(System.in));
39.   System.out.println( "Enter first integer" );
40.   while(!isFull())
41.                           {num=br.readLine();
42.                                push(Integer.parseInt(num));
43.                           }
44.   rev();
45.   while(!isEmpty())System.out.println(pop());
46. }
47. }
```

شرح المثال:

فكرة البرنامج هو استخدام مكدسين آخرين لعملية نقل البيانات فعندما نقلنا البيانات من المكدس الأول إلى المكدس الثاني فان البيانات قد انعكست ولكننا استخدمنا مكدس ثالث لكي نعيد البيانات إلى المكدس الأصلي بالطريقة التي طلبت منا وبالية عمل المكدس .

وهذا الكود يعمل على حذف إي قيمة من المكدس ؟

```
1. // حذف أي عدد من دتخل المكدس
2. import java.io.*;
3. class Stack_Del {
4.   static final int CAPACITY = 5;
5.   static int[] Stack1 = new int[CAPACITY];
6.   static int top = -1;
7.
8.   static boolean isEmpty(){return (top < 0);}
9.
10.   static boolean isFull() {return (top+1== CAPACITY);}
11.
12.   static void push(int element){
13.     if (isFull())
14.                              System.out.println("Stack is full.");
15.                              else
16.                              Stack1[++top] = element;
17.                            }
18.
19.   static int pop() {
20.
21.     if (isEmpty()){
22.                               System.out.println("Stack              is
empty.");
23.                               System.exit( 0 );
24.                              }
25.     return Stack1[top--];
26.                 }
27.   static void delete(int number){ int top2=-1,temp;
28.                                  int[] Stack2 = new int[CAPACITY];
29.                                   while(!isEmpty()){
30.                                              temp=pop();
31.
        if(number!=temp)Stack2[++top2]=temp;
```

```
32.                                         }
33.                                 while(top2>=0)push(Stack2[top2--]);
34.                             }
35.
36. public static void main(String args[])throws IOException {
37.    String num;
38.    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
39.    System.out.println( "Enter first integer" );
40.    while(!isFull())
41.                    {num=br.readLine();
42.                     push(Integer.parseInt(num));
43.                    }
44.
45.    System.out.println("Enter number delete");
46.    num=br.readLine();
47.    delete(Integer.parseInt(num));
48.    System.out.println();
49.    while(!isEmpty())System.out.println(pop());
50.
51. }
52. }
```

1.4.3 الصنف من نوع Stack

لقد بينا في السابق طرق التعامل مع المكدس , حيث قمنا بإنشاء جميع المناهج المتعلقة بعمليات المكدس يدوياً.

لغه Java توفر لك الصنف java.util.Stack الذي يمكنك من التعامل مع حميع عمليات المكدس من حذف,استعادة,حشر إي عنصر من قمة المكدس. ويقدم العديد من المناهج من أجل تقديم بنية معطيات تحقق القاعدة التالية : الداخل أولاً الخارج أخراً LIFI . والجدول 1-15 يبين مناهج هذا الصنف:

جدول 1-15	
empty()	يعيد القيمة true إذا كان المكدس فارغاً
peek()	يعيد العنصر الموجود في قمة المكدس دون أن يحذفه
push()	يضيف عنصر إلى قمه المكدس
search()	يعيد مكان العنصر المحدد ضمن المكدس
remove()	تحذف موقع عنصر ما

ونعرض الآن مثال يستخدم جميع المناهج التي ذكرت في الجدول السابق:

```
1. // استخدام  المكتبة الخاصة بالمكدس
2. import java.io.*;
3. class Stack_All {
4. public static void main(String args[])throws IOException {
5.    String num;
```

```
 6.     int i;
 7.     BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
 8.     java.util.Stack stack= new java.util.Stack();
 9.     System.out.println( "Enter first integer" );
 10.
 11.    for(i=0;i<5;i++)
 12.                    {num=br.readLine();
 13.                     stack.push(new
Integer(Integer.parseInt(num)));
 14.                    }
 15.
 16.    System.out.println("Enter number Select");
 17.    System.out.println("1- Search");
 18.    System.out.println("2- Remove");
 19.    System.out.println("3- Desply");
 20.    num=br.readLine();
 21.
 22.    switch(Integer.parseInt(num)){
 23.            case 1:
 24.                            System.out.println("Enter         number
Search");
 25.                            num=br.readLine();
 26.                            System.out.println("The Pos In "+
 27.                            stack.search(new
Integer(Integer.parseInt(num))));
 28.                            if(stack.search(new
Integer(Integer.parseInt(num)))<0)
 29.                                    System.out.println("Not     Found
");
 30.                            break;
 31.            case 2:
 32.                            System.out.println("Enter  Pos  number
delete");
 33.                            num=br.readLine();
 34.                            stack.remove(Integer.parseInt(num));
 35.
        while(!stack.empty())System.out.print(stack.pop()+" ");
 36.                            break;
 37.            case 3:
 38.
        while(!stack.empty())System.out.print(stack.pop()+" ");
 39.                            break;
 40.                                                    }
 41.
 42. }
 43. }
```

شرح المثال:
في السطر 8 تم اشتقاق صنف جديد باسم stack من المكتبة الخاصة بالتعامل مع المكدس. وفي السطر 13 تم إدخال إلى المكدس العناصر بواسطة التعليمة push , وتلاحظ أننا قمنا بعملية التحوير new Integer لنحدد نوع البيانات التي سيخزنها المكدس.
في السطر 27 نفس قمنا بالبحث عن عنصر . ونلاحظ أن التعليمة search تعطي لنا موقع العنصر إن وجد وإلا تعيد القيمة -1 في حالة إنها لم تجده.
في السطر 34 قمنا بحذف قيمة عنصر بواسطة قيمة الموقع.


من الأخطاء الشائعة إفراغ المكدس بواسطة التعليمة pop ومن ثم نقوم البحث عن عنصر أو حذف عنصر. مما يسبب لنا خطاء في زمن التنفيذ.


عدم استخدام عملية تحديد المعطيات أثناء إدخال القيم للمكدس أو بحث عن قيم , مما يسبب لنا خطاء قواعدي.

1.5 الطوابير (Queues) أو سجلات الآنتظار

وهي عبارة نوع من هياكل البيانات الخطية ويشبه المكدس لتخزين المعلومات بشكل مؤقت مع فارق يكمن في أن التنظيم المتبع لإدخال المعلومات وإخراجها هو FIFO (First Input First Output) إي الداخل أولا الخارج أولا إي تكون عملية الإضافة من النهاية والحذف من الأمام إي يوجد للطابور مؤشرين مؤشر الرأس ويسمى head or front ومؤشر الذيل ويسمى tail or rear وعند الإضافة فإننا نزيد من قيمة الذيل بواحد وعند الحذف فإننا نزيد قيمة الرأس بواحد أيضا فتكون البيانات مرتصة بشكل نتتإلى ومتقاربة على شكل خط وليست على مواقع متفرقة بالذاكرة إي أشبة بالطابور المدرسي فأول طالب حاضر هو أول طالب داخل للفصل .
فهو يشبه طابور الآنتظار للإفراد عند المؤسسة أو المستشفى كما في الشكل 5-15.

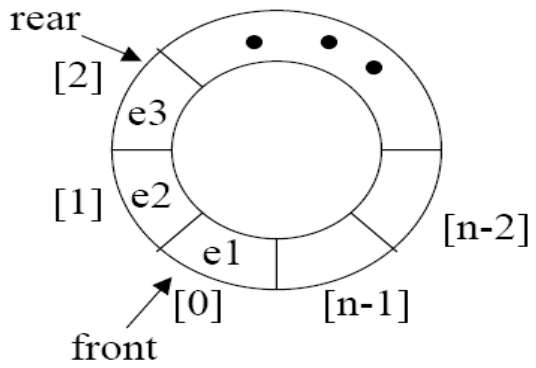
1.5.1 أنواع الطوابير

طابور خطي

وهو له حجم محدود وشرط امتلائه أن تكون قيمة الذيل تساوي حجم المصفوفة .

شكل 5-15

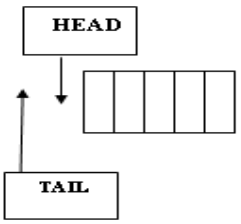
2) طابور دائري / نفس تعريف السابق إلى أن شرط الامتلاء يختلف عن السابق الرأس =1 و الذيل = حجم المتجه أو الرأس=الذيل+1 كما في الشكل 6-15.



شكل 15-6

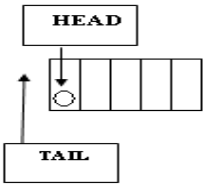
وسنبدأ بالتحدث إلى الطابور الخطي

نجد في البداية يكون الرأس والذيل لا يؤشران لأي موقع ولمعرفة أن الطابور لم تدخل إلية إي قيمة عندما يكون (if(tail==-1||head==-1 والشكل 15-7 يوضح ذلك.



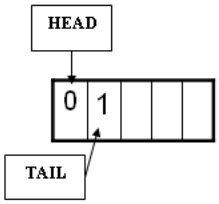
شكل 15-7

وعند إدخال أول قيمة يصبح قيمة الرأس والذيل = 0 كما في الشكل 15-8.



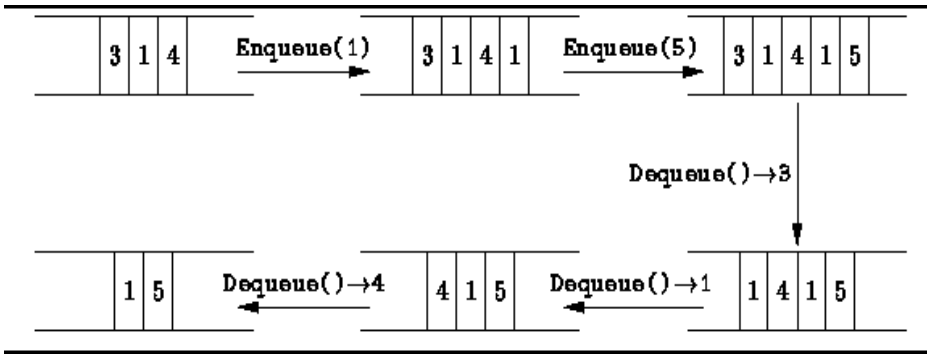
شكل 15-8

وعند إدخال ثاني قيمة نزيد من قيمة الذيل بواحد فقط أم الرأس يبقى كما هو , كما في الشكل 15-9.

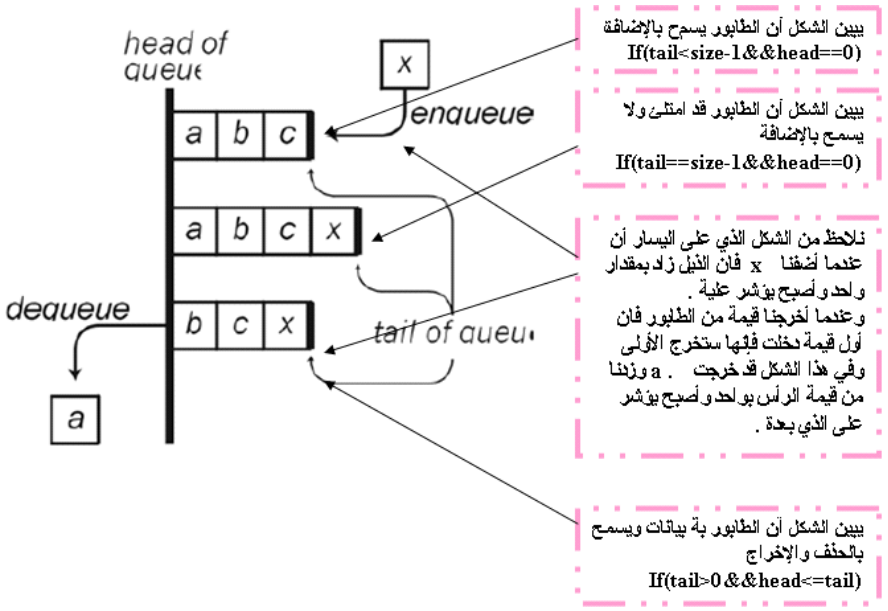


شكل 15-9

وعملية الحذف عكس السابق إي يكون الذيل تابت والرأس يزيد في كل عملية حذف بمقدار واحد مع عمل إزاحة للمتجه لليسار في كل عملية حذف إن أردت .والأشكال 10-15,11-15 تبين عملية الإدخال و الإخراج من داخل الطابور.



شكل 10-15



شكل 11-15

ومن خلال الإشكال السابقة سنورد أول برنامج للطابور

```
1. // برنامج الطابور
2. import java.io.*;
3. class Queue_1 {
4.   static final int CAPACITY = 5;
5.   static int[] Queue = new int[CAPACITY];
6.   static int tail=-1,head=-1;;
7.
8.   static boolean isEmpty(){return (tail < 0||head>tail);}
```

```
 9.
 10.   static boolean isFull() {return (tail== CAPACITY-1);}
 11.
 12.   static void add_Queue(int element){
 13.    if (isFull())
 14.                          System.out.println("Is FULL Queue");
 15.                          else
 16.                          {
 17.                           if(tail==-
1){head=tail=0;Queue[tail]=element;}
 18.                                 else
 19.                                 Queue[++tail]=element;
 20.                          }
 21.                                                  }
 22.
 23.   static int De_Queue() {
 24.
 25.    if (isEmpty()){
 26.                           System.out.println("Queue          is
empty.");
 27.                           System.exit( 0 );
 28.               }
 29.    return Queue[head++];
 30.                            }
 31.
 32. public static void main(String args[])throws IOException {
 33.   String num;
 34.   BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
 35.   System.out.println( "Enter first integer" );
 36.   while(!isFull())
 37.                          {num=br.readLine();
 38.
       add_Queue(Integer.parseInt(num));
 39.                          }
 40.
 41.   System.out.println();
 42.   while(!isEmpty())System.out.print(De_Queue()+" ");
 43.
 44.
}
 45. }
```

1.5.2 العمليات على الطابور

*)الإضافة ADD

والكود التابع لهذه العملية هو نفس الكود السابق

*)الحذف DEL
وهذا الكود لهذه العملية

```
// برنامج الحذف من الطابور
import java.io.*;
class Chp13_2 {
  static final int CAPACITY = 5;
  static int[] Queue = new int[CAPACITY];
  static int tail=-1,head=-1;

  static boolean isEmpty(){return (tail < 0||head>tail);}

  static boolean isFull() {return (tail== CAPACITY-1);}

  static void add_Queue(int element){
    if (isFull())
                System.out.println("Is FULL Queue");
                else
                {
                 if(tail==-1){head=tail=0;Queue[tail]=element;}
                   else
                   Queue[++tail]=element;
                }
                            }

  static int De_Queue() {

    if (isEmpty()){
                  System.out.println("Queue is empty.");
                  System.exit( 0 );
                 }
    return Queue[head++];
                  }

   static void del_Queue(int element){
     int[] Queue2 = new int[CAPACITY];
     int tail2=-1,head2=-1;
     if(isEmpty()){
                    System.out.println("Queue is empty.");
                    System.exit( 0 );
                  }
     else
     while(head<=tail){
                     if(Queue[head]!=element){
                     if(tail2==-1){head2=tail2=0;Queue2[tail2
                      else
                      Queue2[++tail2]=Queue[head];
                                                 }
                     head++;
                    }
     head=tail=-1;
     while(tail2>=head2){
                    if(tail==-1){head=tail=0;Queue[tail]=Queu
                    else
                    Queue[++tail]=Queue2[head2];
     head2++;             }
                                    }

public static void main(String args[])throws IOException {
     String num;
     BufferedReader br = new BufferedReader(new InputStreamR
     System.out.println( "Enter first integer" );
     while(!isFull())
                     {num=br.readLine();
                      add_Queue(Integer.parseInt(num));
                     }
     System.out.println("Enter number delete");
     num=br.readLine();
     del_Queue(Integer.parseInt(num));

     System.out.println();
     while(!isEmpty())System.out.print(De_Queue()+" ");

}
}
```

شرح المثال:
الفكرة المستخدمة بالبرنامج هو خلق طابور جديد وإدخال جميع القيم ماعدا القيمة التي تساوي القيمة المراد حذفها تم نقل الطابور الجديد للقديم كما في الأسطر من 40 إلى 53.

*) البحث
نفس البرنامج السابق إلى إننا لا نقوم بخلق طابور ونقل بل إننا نبحث علية إن وجد نطبعه وإلا نطبع إننا لم نحصل علية . كما يلي:

```
1. // برنامج البحث عن عنصر بداخل الطابور
2. import java.io.*;
3. class Queue_Search {
4.   static final int CAPACITY = 5;
5.   static int[] Queue = new int[CAPACITY];
6.   static int tail=-1,head=-1;
7.
8.   static boolean isEmpty(){return (tail < 0||head>tail);}
9.
10.   static boolean isFull() {return (tail== CAPACITY-1);}
11.
12.   static void add_Queue(int element){
13.     if (isFull())
14.                             System.out.println("Is FULL Queue");
15.                             else
16.                             {
17.                              if(tail==-
1){head=tail=0;Queue[tail]=element;}
18.                                     else
19.                                     Queue[++tail]=element;
20.                             }
21.                           }
22.
23.   static int De_Queue() {
24.
25.     if (isEmpty()){
26.                         System.out.println("Queue is empty.");
27.                         System.exit( 0 );
28.                 }
29.     return Queue[head++];
30.                                     }
31.
32.   static void F_Queue(int element)
33.             {int y=0,temp;
34.                          if (isEmpty()){
35.                             System.out.println("Queue                is
empty.");
36.                             System.exit( 0 );
```

```
37.                         }
38.             else
39.             while(!isEmpty()){temp=De_Queue();
40.                                     if(temp==element)
41.                                         {y=1;
42.
System.out.println("FOUND "+temp);
43.                                          break;
44.                                         }
45.                             }
46.             if(y==0)System.out.println("NOT                FOUND
"+element);
47.             }
48.
49.
50. public static void main(String args[])throws IOException {
51.    String num;
52.    BufferedReader    br    =    new    BufferedReader(new
InputStreamReader(System.in));
53.    System.out.println( "Enter first integer" );
54.    while(!isFull())
55.                             {num=br.readLine();
56.
       add_Queue(Integer.parseInt(num));
57.                             }
58.    System.out.println("Enter number Search");
59.    num=br.readLine();
60.    F_Queue(Integer.parseInt(num));
61.
}
62. }
```

*) دمج طابورين

سأذكر فكرة البرنامج وعلى القارئ أن يحل الكود

أولا يجب التأكد من أن الطابور الثاني بوجد فيه مساحة كافية لاستيعاب قيم الطابور الأول ومن العلاقة الآتية

(عدد العناصر الموجودة في الأساسي-عدد القيم)=>(عدد العناصر بداخلة - حجم الطابور المستضيف) ثم بعد ذلك نضيف الطابور الثاني في الأول .

الطابور الدائري

نفس البرامج التي ذكرناها سابقاً والاختلاف سيكون في أوامر الشرط

يكون الطابور فارغاً if(head==tail+1) .

يكون الطابور غير ممتلئ if(tail!=size&&head=1) .

يكون الطابور ممتلئا if(head==1&&tail==size).

ونفس البرامج التي ذكرتها بالمكدس تطبق على الطوابير بالية الطابور.

وإلى هنا يجب على القارئ أن يكون قد اتضحت فكرة الطابور .

1.6 القوائم (List)

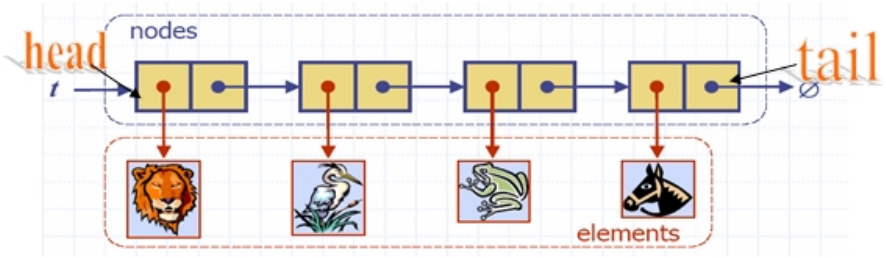
وهي نوع من هياكل البيانات الخطية تتألف من مجموعة من الخلايا المرابطة فيما بينها وكل عنصر فيها يسمى عقدة وهذه العقدة فيها حقلين حقل للقيم وحقل يؤشر لعنوان للعقدة الذي بعدها أو قبلها أو NULL وتستعمل هذه الكلمة للدلالة إلى نهاية اللائحة , ومن الممكن أن تتألف العقدة على أكثر من مؤشر ومعلومات إي قيم , فتكون ضمن مجموعة (block) أو كتلة , ولا بد من مؤشر يؤشر إلى أول عقدة ومؤشر يؤشر إلى أخر عقدة إي مثل الطابور .

أنواع القوائم

- القوائم الأحادية.
- القوائم المذبلة أي الثنائية.
- القوائم الدائرية.

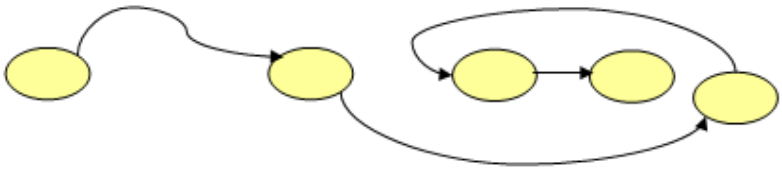
1.6.1 القوائم الأحادية

تشبه حبل الغسيل تعلق علية البيانات تتالياً إذا كان الإدخال في نفس الوقت ويوجد عنوان راسي يؤشر إلى أول عنصر من اللائحة ويسمى head ويوجد عنوان نهائي يؤشر إلى أخر عنصر من اللائحة ويسمى tail وكل عقدة تؤشر إلى العقدة التالية وأخر عقدة تكون قيمة المؤشر لها NULL و تكون كما في الشكل 12-15



شكل 12-15

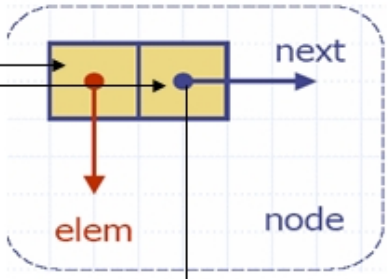
وليس من الضروري أن تكون العقد مرتبة بشكل متتالي في الذاكرة فهي تكون مبعثرة في الذاكرة لان الجهاز الذي يحجزها في الذاكرة وليس اليوزر لكنها متصلة فيما بينها بواسطة المؤشرات و الشكل 13-15 يبين كيف تكون شكلها .



شكل 13-15

- كيفية تعريف الهيكل العام للعقد

```
class Node {
  element;
  Node next;
  Node( s) {
              element = s;
              next = null;
             }
getElement( ) { return element; }
void setElement(newElem) { element = newElem; }
}
```



شكل 14-15

ما معنى هذا الحقل next ؟

معناه مؤشر من نوع الصنف Node نفسه , أي يؤشر إلى صنف Node أخر من نفس النوع , أي نستطيع الوصول إلى Node أخر وأخر وهكذا إلى مالا نهاية. ويكون بداخل العقدة موقع العقدة التي بعدها أو قبلها.

أي لنتذكر أحبائي علبة الحليب حيث أن بداخلها نفس صورة العلبة نفسها و بداخل الصورة نفس الصورة العلبة ووو إلا مالا نهاية . والعقدة هنا نفس الشيء حيث أن داخلة عقدة وبداخل العقدة عقدة وهكذا .

المنهج getElement() :

تعطي لنا قيمة العقدة.

المنهج setElement () :

نخزن قيمة مرسلة للعقدة.

وللإضافة عدة أنواع

- الإضافة من اليمين
- الإضافة من اليسار
- الإضافة من أي مكان

وهذا أول مثال لهذه القوائم وهو الإضافة من اليمين للقائمة:

```
1. // برنامج الأضافة من اليمين للقائمة الأحادية
2. class List_add{
3. public static void main(String args[]){
4.   Node head=null;              // head node of the list
5.   Node tail=null;              // tail node of the list
6.   Node node=null;
7.    int size=5;
8.
9. tail=head=node=new Node(0);
10.  for (int i=1;i<size;i++)
11.  {
12.   node=new Node(i);
13.   tail.setNext(node);
14.   tail=node;
15.  }
16.
```

```
17.  node=head;
18.
19.  while(node!=null)
20.  {
21.   System.out.print(node.getElement()+" ");
22.   node=node.getNext();
23.  }
24.  System.out.println();
25. }
26. }
27.
28. /** Node of a singly linked list of ints. */
29. class Node {
30.   private int element;// we assume elements are character ints
31.   private Node next;
32.   /** Creates a node with the given element and next node. */
33.   public Node(int s) {
34.     element = s;
35.     next = null;
36.   }
37.   /** Returns the element of this node. */
38.   public int getElement() { return element; }
39.   /** Returns the next node of this node. */
40.   public Node getNext() { return next; }
41.   // Modifier methods:
42.   /** Sets the element of this node. */
43.   public void setElement(int newElem) { element = newElem; }
44.   /** Sets the next node of this node. */
45.   public void setNext(Node newNext) { next = newNext; }
46. }
```

شرح المثال:

سنبدأ بشرح المناهج المضافة للصنف Node :

السطر 38 منهج يعيد لنا قيمة العقدة.

السطر 40 منهج يعيد لنا موقع العقدة التالية.

السطر 43 منهج يخزن قيمة مرسلة للعقدة في المتحول element.

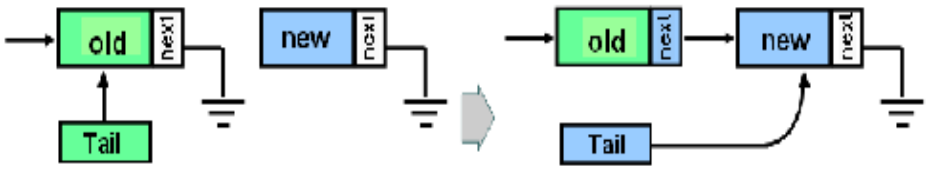
السطر 45 منهج يخزن موقع عقدة مرسلة للعقدة في المتحول next.

الأسطر (4 - 6) هنا عرفنا head من نوع Node مؤشر لعقدة وهو الرأس و tail من نوع Node مؤشر لعقدة وهو الذيل و node سنستخدمه كمتغير لإدخال بيانات العقد .

قم بإسناد القيمة null إلى التوابع الخاصة بالعقدة لأنة يجب إعطاء قيم ابتدائية لها قبل التعامل معها.

السطر 9 تم إنشاء أول عقدة وإرسال إليها القيمة 0 , وجعل العقدة تساوي head , tail , node حيث تعتبر هذه أهم مرحلة عند إنشاء العقد.

عند إنشاء العقد يجب أن تنشئ أول عقدة بمفردها حتى يتم مساواة الرأس والذيل بها.
الأسطر (10 - 15) هنا سنكوّن 4 عقد أضافية بجانب الأولى فيكون لدينا 5 عقد .
السطر 13 يبين أن حقل العقدة الأولى يساوي العقدة الجديدة .فبهذه الحالة تمت عملية الربط بين العقدتين بقي علينا نقل الذيل إلى العقدة الجديدة tail=node كما في السطر 14 .
وهكذا بباقي العقد إلى أن ينتهي عمل اللوب ويمكنك تكوين مئات العقد بهذه الطريقة والشكل 15-15 يبين الشرح .
الأسطر (17 - 23) في عملي طباعة العقد أول شي يجب أن تعمله هو الوصول لأول عقدة فكيف ستعمل لو تتذكر قليل أن أول ما أنشانا أول عقدة ساوينا الرأس والذيل بها وبعد ذلك كان كل ما أضفنا عقدة جديدة تحرك معانا الذيل وأصبح الذيل بمؤخرة العقد والرأس في بداية العقد .إذن node= head فنكون وصلنا إلى أول عقدة كما في السطر17 .
بقي علينا طباعة العقد والتنقل إلى العقدة التالية كما في السطر 22. بمعنى أن العقدة التي واقفين عليها تساوي حقل العقدة نفسها التي داخلة موقع العقدة التالية فبذلك نكون قد انتقلنا إلى العقدة التالية وتستمر هذه العملية إلى أن تساوي العقدة NULL فينتهي عمل اللوب .



لشكل 15-15

أم الإضافة من اليسار نفس المثال السابق إلا أن الاختلاف فقط بعملية إدخال العقد الثانية وما بعدها

```
for (int i=1;i<size;i++)
{
 node=new Node(i);
 node.setNext(head);
 head=node;
}
```

الإضافة من اليمين يكون الرأس متحرك والإضافة من اليمن يكون الذيل هو المتحرك .

وهذا مثال على إضافة عقدة بعد قيمة عقدة يريدها المستخدم :

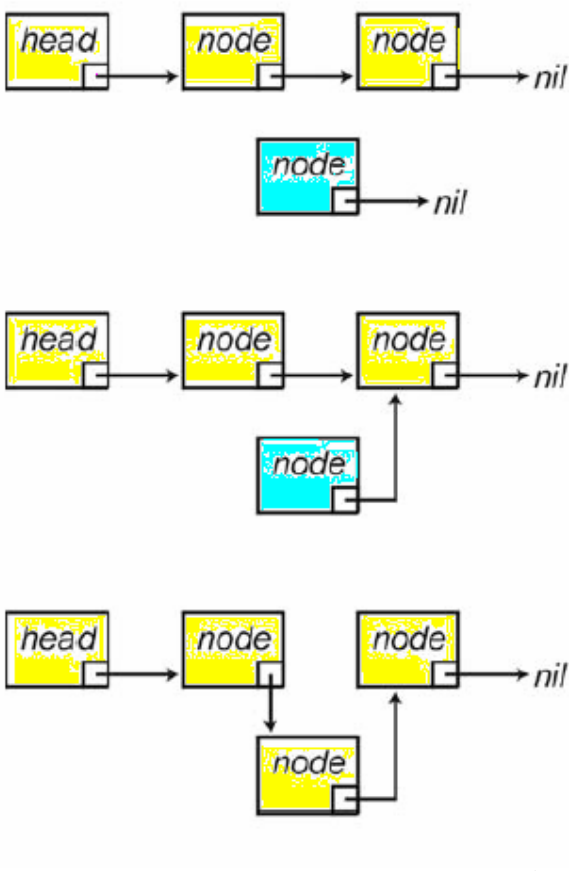
```
1. // برنامج الأضافة بعد قيمة عقدة معينة
2. import javax.swing.JOptionPane;
3. class Raed1 extends  Node{
4. public static void main(String args[]){
5.   Node head=null;              // head node of the list
6.   Node tail=null;              // tail node of the list
7.   Node node=null;
8.    int size=5;
9.
```

```
10.  tail=head=node=new Node(0);
11.  for (int i=1;i<size;i++)
12.  {
13.   node=new Node(i);
14.   node.setNext(head);
15.   head=node;
16.  }
17.
18.  node=head;
19.
20.  while(node!=null)
21.  {
22.   System.out.print(node.getElement()+" ");
23.   node=node.getNext();
24.  }
25.  System.out.println();
26.
27.    String snum1;
28.    int num1;
29.    snum1 = JOptionPane.showInputDialog("Enter num1:") ;
30.    num1 = Integer.parseInt(snum1);
31.
32.  node=head;
33.  while(node!=null)
34.  {
35.    if(node.getElement()==num1)
36.                              {
37.                               Node temp;
38.                               snum1=JOptionPane.showInputDialog

("Enter Value  node:") ;
39.                                        num1                                =
Integer.parseInt(snum1);
40.                               temp=new Node(num1);
41.                               temp.setNext(node.getNext());
42.                               node.setNext(temp);
43.                               break;
44.                              }
45.    node=node.getNext();
46.  }
47.  node=head;
48.  while(node!=null)
49.  {
50.   System.out.print(node.getElement()+" ");
51.   node=node.getNext();
52.  }
```

```
53.  System.out.println();
54.  System.exit(0);
55. }
56. }
```

شرح المثال:
بعد إدخال العقد طلبنا من المستخدم إدخال قيمة فإذا وجدت هذه القيمة بالقائمة سنضع العقدة الجديدة بعدها مباشرة .
فعملنا عملية بحث عن العنصر إذا وجد فإننا سنعمل على إنشاء عقدة جديدة وسندخل قيمة العقدة وسنربط حقل مؤشر العقدة الجديدة بالعقدة التي بعد العنصر والعقدة التي مازلنا واقفين عليها تم ربط مؤشرها بالعقدة الجديدة كما في الأسطر (35 - 44) والشكل 16-15 يبين هذه العملية .



شكل 16-15

1.6.2 صنع المكدسات و الطوابير ديناميكياً

تحدثنا عن الهياكل الإستاتيكيه أي الثابتة وتكلمنا عن المكدسات والطوابير لنعمل على تطبيق تلك الخوارزميات بالقوائم الأحادية ونجعلها متغيرة أي ديناميكية ونتخلص من شيء أسمة المكدس قد امتلئ أو الطابور قد امتلئ والآن سنورد مثال عن المكدس باستخدام القوائم الأحادية والية الإدخال والإخراج قد تكلمنا عنها في السابق .
واليكم الكود :

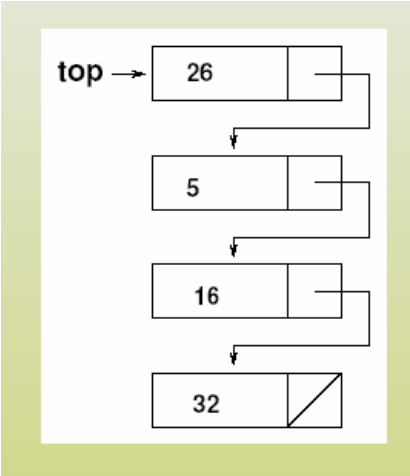
```
1. // برنامج مكدس بواسطة القوائم
2. class Raed2 extends  Node{
3. public static void main(String args[]){
4.    int size=5;
5.  stack stack1=new stack();
6.  for (int i=1;i<size;i++)
7.    stack1.push(i);
8.
9.  while(!stack1.isEmpty())
10.   System.out.print(stack1.pop()+" ");
11.
12.  System.out.println();
13. }
14. }
15.
16. /** A linked Stack. */
17. class stack extends Node{
18.
19.   public Node Stack1=null,top =null ;
20.   //* Return whether the stack is empty.
21.   public boolean isEmpty(){return (top == null);}
22.   //*Insert an element at the top of the stack.
23.   public void push(int element){
24.  if(top==null)
25.                     {
26.                      top=Stack1=new Node(element);
27.                     }
28.                     else
29.                     {
30.                      Stack1=new Node(element);
31.                      Stack1.setNext(top);
32.                      top=Stack1;
33.                               }
34.                                       }
35.     /**
36.     * Remove the top element from the stack.
37.     * @return element removed.
38.     * @exception EmptyStackException if the stack is empty.
39.     */
40.    public int pop() {
41.
42.     if (isEmpty()){
43.                                System.out.println("Stack is empty.");
44.                                System.exit( 0 );
45.                               }
46.                                int temp=top.getElement();
```

```
47.     top = top.getNext();
48.     return temp;
49.                             }
50. }
```

الشكل 15-17 يبين شكل العقد للمكدس

وجميع ما ذكرناه من تطبيقات المكدس والطوابير على القارئ أن يطبق تلك الأمثلة بالقائمة الأحادية.

وبعد أن تكلمت على عملية الإضافة بجميع أنواعها بالقائمة الأحادية يبقى لنا الآن أن نتكلم عن عملية الحذف وما يدور من تطبيقات حولها.



شكل 15-17

- الحذف (DELETE)

هو عملية بسيطة في لائحة الوصل الخطية , فتوصل وصلة العنصر الذي يأتي قبل العنصر المراد حذفه بعنوان العنصر الذي يراد حذفه فتعتبر عملية عكسية لعملية الإضافة .

وللحذف عدة أنواع

- الحذف من الرأس
- الحذف من الذيل
- الحذف من أي مكان

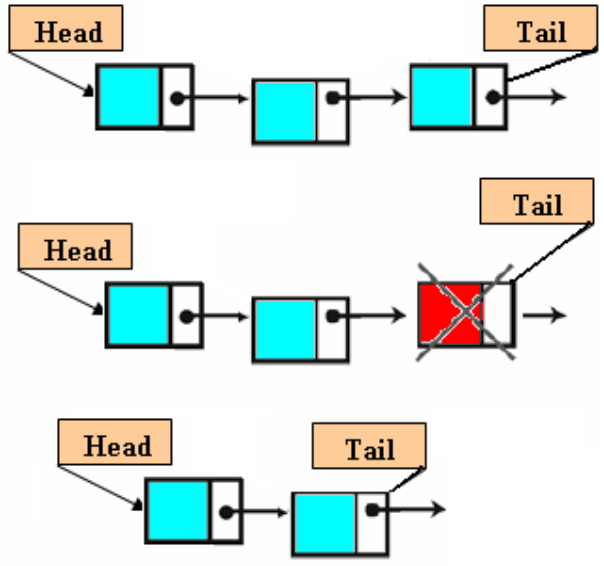
وسنرى أول مثال لهذه القوائم وهو الحذف من النهاية للقائمة أي أخر عقدة.

ويتم ذلك جعل العقدة قبل الأخيرة في القائمة مساوية NULL تم ننقل الذيل إلى وراءه بمقدار واحد أي العقدة التي قبل الأخير ثم نحذف العقدة الأخيرة . كما في الشكل 15-18.

وهذا المثال لهذه العملية:

```
1. // برنامج لحذف عقدة من نهاية القائمة الأحادية
2. class Raed3 extends  Node{
3. public static void main(String args[]){
4.   Node head=null;              // head node of the list
5.   Node tail=null;              // tail node of the list
6.   Node node=null;
7.    int size=5;
8.
9. tail=head=node=new Node(0);
10.  for (int i=1;i<size;i++)
11.  {
12.   node=new Node(i);
13.   tail.setNext(node);
14.   tail=node;
15.  }
16.
17.  node=head;
18.
19.  while(node!=null)
20.  {
```

```
21.   System.out.print(node.getElement()+" ");
22.   node=node.getNext();
23.  }
24.  System.out.println();
25.
26.  node=head;
27.  while(node!=null)
28.  {
29.    if(node.getNext()==tail)
30.                              {
31.                               tail=null;
32.                               node.setNext(null);
33.                               tail=node;
34.                               break;
35.                              }
36.    node=node.getNext();
37.  }
38.  node=head;
39.  while(node!=null)
40.  {
41.   System.out.print(node.getElement()+" ");
42.   node=node.getNext();
43.  }
44.  System.out.println();
45. }
46. }
```



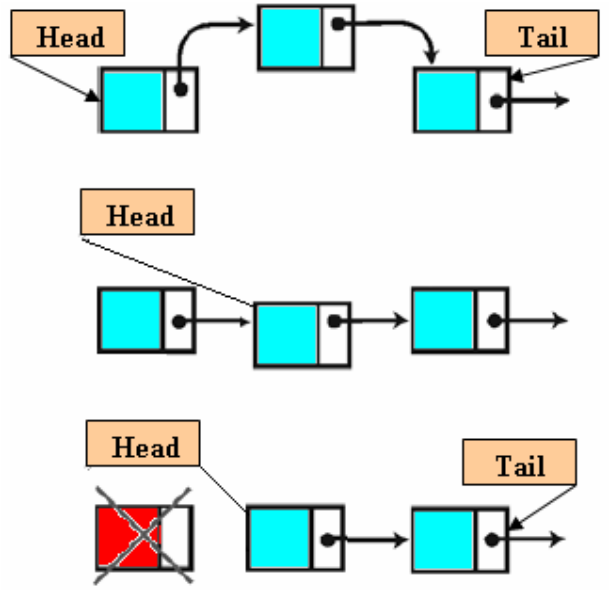
شكل 15-18

واليكم هذا الكود لعملية الحذف من الإمام أي أول عقدة .
سننقل الرأس إلى الإمام بمقدار واحد أي للعقدة التي بعدها ثم نحذف أول عقدة . كما في الشكل 15-19.

وهذا المثال لهذه العملية:

```
1. // برنامج لحذف أول عقدو من داخل القائم الأحادية
2. class Raed4  extends  Node{
3. public static void main(String args[]){
4.   Node head=null;                // head node of the list
5.   Node tail=null;                // tail node of the list
6.   Node node=null;
7.    int size=5;
8.
9. tail=head=node=new Node(0);
10.  for (int i=1;i<size;i++)
11.  {
12.   node=new Node(i);
13.   tail.setNext(node);
14.   tail=node;
15.  }
16.
17.  node=head;
18.
19.  while(node!=null)
20.  {
21.   System.out.print(node.getElement()+" ");
22.   node=node.getNext();
23.  }
24.  System.out.println();
25.
```

```
26.  node=head;
27.  head=node.getNext();
28.  node=null;
29.
30.  node=head;
31.  while(node!=null)
32.  {
33.   System.out.print(node.getElement()+" ");
34.   node=node.getNext();
35.  }
36.  System.out.println();
37. }
38. }
```



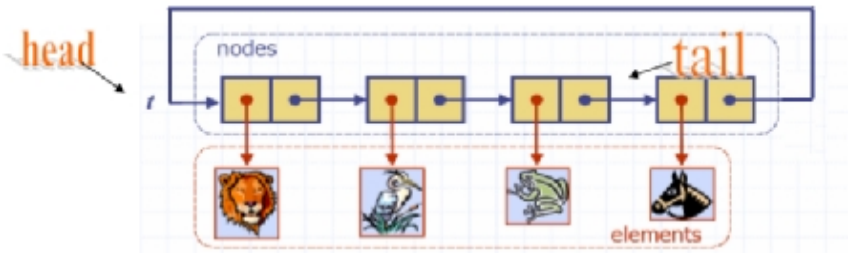
وعمليه الحذف من الوسط يتم البحث عن العقدة المراد حذفها تم نغير حقل المؤشر للعقدة الذي قبلها بالعقدة التي بعدها ويتم ذلك بجعل مؤشر يمشي ورآنا بمقدار واحد بعملية جعل متغير وهذا المتغير ياخد قيمه العقدة تم ننتقل للعقدة التي بعدها .

شكل 15-19

1.6.3 القوائم الأحادية المتصلة

حيث يشير مؤشر العقدة الأخيرة إلى العقدة الأولى أي مؤشر الذيل سيؤشر إلى الرأس والشكل 15-20 يوضح ذلك.

وتستعمل هده القوائم المتصلة كثيراً في أنظمة إدارة بنوك المعطيات وفي البرمجة إذ تسمح بربط العناصر التي تتمتع بنفس الخصائص فيما بينها .

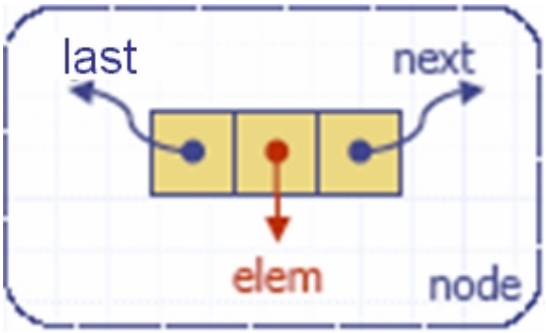


شكل 15-20

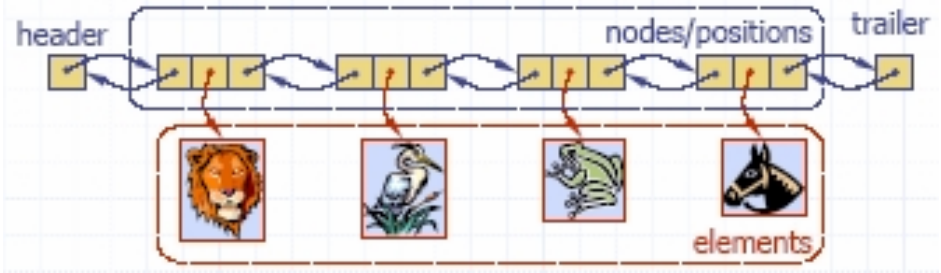
وعندما تريد تحويل القائمة الدائرية إلى قائمة الخطية نجعل مؤشر أي عقدة في القائمة مساوياً إلى (NULL) فتتحول إلى قائمة متصلة .

1.6.4 القوائم المذبلة الثنائية

تعتبر القوائم الثنائية قوائم أحاديه ولكن ليس العكس حيث أن القوائم المذبلة لها مؤشرين مؤشر يؤشر إلى العقدة التالية ويسمى next ومؤشر يشير إلى العقدة السابقة يسمى last .
وتستعمل هذه القوائم عندما نحتاج للرجوع إلى وراء لجلب معلومات معينة ولنتذكر برنامج معالجة النصوص حيث أنة يستطيع العودة إلى الورى لتعديل حرف مثلاً.
ويكون الهيكل العام لها كما في الشكل 21-15 والشكل 22-15 يبين الشكل العام للقوائم المذبلة.



شكل 21-15



شكل 22-15

ومن الشكل 22-15 يتضح لنا شكل هذه القوائم . وسنقوم الآن بإنشاء صنف يمثل القوائم المذبلة:

```
1. /**
2.  * Class binary tree by storing references to
3.  * an element, a parent node, a left node, and a right node.
4.  */
5. public class BTNode{
6.   private int element;                    // element stored at this
node
7.   private BTNode left, right;  // adjacent nodes
8.   /** Main constructor */
9.   public BTNode(){}
10.
11.   public BTNode(int element) {
12.     setElement(element);
13.     setLeft(null);
14.     setRight(null);
15.                                                  }
16.   /** Returns the element stored at this position */
```

```
17.  public int element() { return element; }
18.  /** Sets the element stored at this position */
19.  public void setElement(int o) { element=o; }
20.  /** Returns the left child of this position */
21.  public BTNode getLeft() { return left; }
22.  /** Sets the left child of this position */
23.  public void setLeft(BTNode v) { left=v; }
24.  /** Returns the right child of this position */
25.  public BTNode getRight() { return right; }
26.  /** Sets the right child of this position */
27.  public void setRight(BTNode v) { right=v; }
28. }
```

شرح المثال:
جميع المناهج في الصنف DNODE هي نفسها في صنف NODE في القوائم الأحادية. الاختلاف فقط هو المتحول last وهو نفس المتحول next الذي يؤشر إلى العقدة السابقة.

نفس العمليات التي طبقت على القوائم الأحادية ستطبق على القوائم المذبلة.

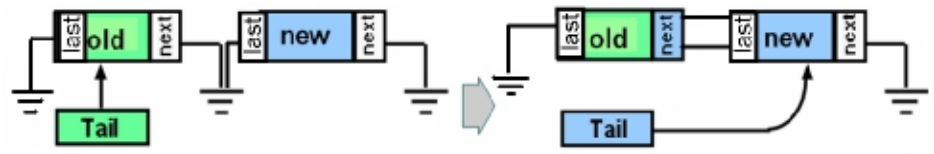
1.6.5 العمليات على القوائم المذبلة

ونبدأ بأول عمليه ألا وهي عملية الإضافة من اليمين.
وهذا المثال لهذه العملية :

```
1. // برنامج الأضافة من اليمين للقائمة الثنائية
2. class Raed5 extends DNODE {
3. public static void main(String args[]){
4.   DNODE head=null;          // head DNODE of the list
5.   DNODE tail=null;          // tail DNODE of the list
6.   DNODE node=null;
7.    int size=5;
8.
9.    tail=head=node=new DNODE(0);
10.  for (int i=1;i<size;i++)
11.  {
12.   node=new DNODE(i);
13.   tail.setNext(node);
14.   node.setPrev(tail);
15.   tail=node;
16.  }
17.
18.  node=head;
19.
20.  while(node!=null)
21.  {
22.   System.out.print(node.getElement()+" ");
```

```
23.   node=node.getNext();
24.  }
25.  System.out.println();
26.
27.  node=tail;
28.  while(node!=null)
29.  {
30.   System.out.print(node.getElement()+" ");
31.   node=node.getPrev();
32.  }
33.  System.out.println();
34. }
35. }
```

شرح المثال:
الأسطر (4-5) عرفنا head من نوع DNODE مؤشر لسجل وهو الرأس و tail من نوع DNODE مؤشر لسجل وهو الذيل و node سنستخدمه كمتغير لإدخال بيانات العقد .
السطر 9 خطوة ضرورية ولابد أن تكون منفردة عن أخواتها لكي نساوي الرأس والذيل بأول عقدة .
الأسطر (16 - 10) سنكون 4 عقد أضافية بجانب الأولى فيكون لدينا 5 عقد .
السطر 13 تحويل حقل الذيل من null إلى عنوان العقدة الجديدة.
السطر 14 حقل العقدة الثانية يساوي العقدة القديمة إي الذيل.
فبهذه الحالة تمت عملية الربط بين العقدتين بقي علينا نقل الذيل إلى العقدة الجديدة tail=node كما في السطر 15 وهكذا بباقي العقد إلى أن ينتهي عمل اللوب ويمكنك تكوين مئات العقد بهذه الطريقة والشكل 23-15 يبن الشرح .



شكل 23-15

أم الإضافة من اليسار نفس السابق إلا أن الاختلاف فقط بعملية إدخال العقد الثانية وما بعدها كهذه الشفرة :

```
for (int i=1;i<size;i++)
{
 node=new DNODE(i);
 node.setNext(head);
 head.setPrev(node);
 head=node;
}
```


الإضافة من اليمين يكون الرأس متحركا والإضافة من اليمن يكون الذيل هو المتحرك.

ونفس البرامج التي ذكرناها في القوائم الأحادية تطبق على القوائم الثنائية فلاختلاف فقط هو زيادة المؤشر الخلفي وربطة بالعقدة الجديدة.

وهذا المثال لعملية ترتيب قائمة ثنائية

```
1. // برنامج ترتيب لقائمة الثنائية
2. class Raed6 extends DNODE {
3. public static void main(String args[]){
4.   DNODE head=null;          // head DNODE of the list
5.   DNODE tail=null;          // tail DNODE of the list
6.   DNODE node=null;
7.    int size=5;
8.
9.    tail=head=node=new DNODE(0);
10.  for (int i=1;i<size;i++)
11.  {
12.   node=new DNODE(i);
13.   tail.setNext(node);
14.   node.setPrev(tail);
15.   tail=node;
16.  }
17.
18.  node=head;
19.
20.  while(node!=null)
21.  {
22.   System.out.print(node.getElement()+" ");
23.   node=node.getNext();
24.  }
25.  System.out.println();
26.
27.  DNODE temp,temp2;
28.
29.  for(temp=head;temp!=null;temp=temp.getNext())
30.    for(temp2=head;temp2!=null;temp2=temp2.getNext())
31.           if(temp.getElement()>temp2.getElement())
32.                  {
33.                   int j;
34.                   j=temp2.getElement();
35.                   temp2.setElement(temp.getElement());
36.                   temp.setElement(j);
37.                  }
38.
39.  node=head;
40.
41.  while(node!=null)
42.  {
```

```
43.   System.out.print(node.getElement()+" ");
44.   node=node.getNext();
45.  }
46.  System.out.println();
47. }
48. }
```

شرح المثال:
السطر 27 عرفنا متغيرين من نوع DNODE وتعاملنا بعملية الترتيب كترتيب مصفوفة وهذه الخوارزمية معروفة ولا جديد فيها.

ينبغي على القارئ حل هذا المثال بدون أن ينضر للكود المكتوب

- اكتب برنامج يعمل على إدخال الأعداد الفردية من اليسار و الزوجية من اليمين؟

هل اكتشفت فكرة البرنامج فهي سهلة جداً ولا تحتاج إلى جهد وضياع للوقت !
إن لم تتضح لك الفكرة يا عزيزي فصلي على معدن الأسرار ومنبع الآنوار سيدنا محمد وعلى آله وصحبة الأطهار وتتبع هذا الكود.

```
1. /*برنامج الأضافة من اليمين للقائمة الثنائية
2. الأعداد الزوجية ومن اليسار الأعداد الفردية
3. */
4. class raed_7 extends DNODE {
5. public static void main(String args[]){
6.   DNODE head=null;          // head DNODE of the list
7.   DNODE tail=null;          // tail DNODE of the list
8.   DNODE node=null;
9.    int size=5;
10.
11.  tail=head=node=new DNODE(0);
12.  for (int i=1;i<size;i++)
13.  {
14.   if(i%2==0){//if number evn
15.                  node=new DNODE(i);
16.                  tail.setNext(node);
17.                  node.setPrev(tail);
18.                  tail=node;
19.             }
20.             else
21.             {//if number add
22.                  node=new DNODE(i);
23.                  node.setNext(head);
24.                  head.setPrev(node);
25.                  head=node;
26.             }
27. }
28.    //print DNODE
29.  node=head;
```

```
30.  while(node!=null)
31.  {
32.   System.out.print(node.getElement()+" ");
33.   node=node.getNext();
34.  }
35.  System.out.println();
36.
37. }
38. }
```

شرح المثال:
فكرة البرنامج هي بعد إنشاء أول عقدة يتم إنشاء ثاني عقدة ويتم تفحص القيمة فإذا كانت زوجيه فان الإضافة ستكون من اليمين وإلا ستكون الإضافة من اليسار. كما في الأسطر (14 - 26).

عمليات الحذف
كما نفذت عملية الحذف في اللوائح الأحادية, هي نفسها تنفذ في اللوائح الثنائية .
الشيء الذي نريد توضيحه هو عندما يراد منك حذف عقدة من أي مكان مع الاحتفاظ برأس اللائحة وذيل اللائحة .

هذا مثال يعمل على حذف إي عقدة بالقائمة

```
1. // برنامج حذف عقدة من القائمة الثنائية
2. import javax.swing.JOptionPane;
3. class raed_8 extends DNODE {
4. public static void main(String args[]){
5.   DNODE head=null;          // head DNODE of the list
6.   DNODE tail=null;          // tail DNODE of the list
7.   DNODE node=null;
8.    int size=5;
9.
10.  tail=head=node=new DNODE(0);
11.  for (int i=1;i<size;i++)
12.  {
13.   node=new DNODE(i);
14.   tail.setNext(node);
15.   node.setPrev(tail);
16.   tail=node;
17.  }
18.
19.    //print DNODE
20.  node=head;
21.  while(node!=null)
22.  {
23.   System.out.print(node.getElement()+" ");
24.   node=node.getNext();
25.  }
```

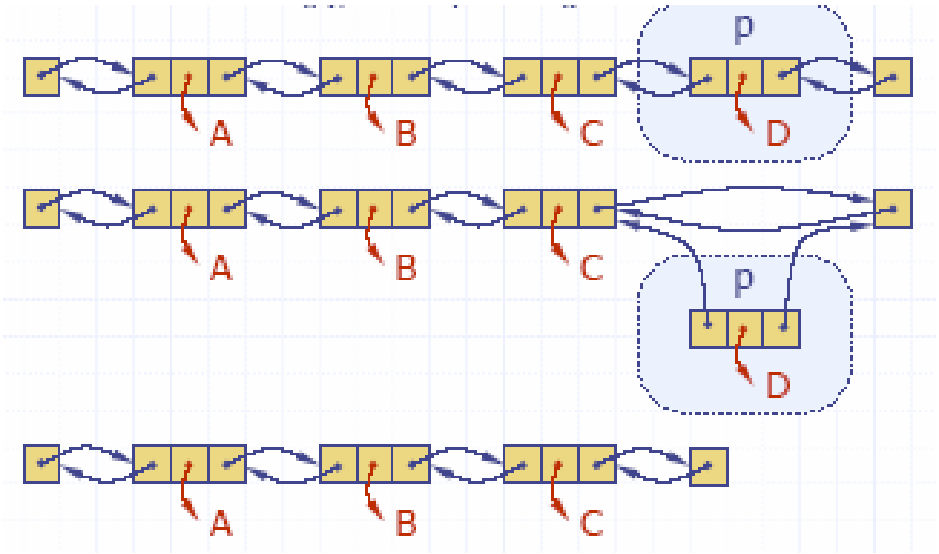
```
26.  System.out.println();
27.
28.  String snum1;
29.    int num1;boolean flag=false;
30.    snum1    =    JOptionPane.showInputDialog("Enter    Number
Delete:") ;
31.    num1 = Integer.parseInt(snum1);
32.    node=head;
33.  while(node!=null)
34.  {
35.    if(node.getElement()==num1)
36.                              {
37.                               if(node==head)
38.                                                    {
39.
node.getNext().setPrev(null);
40.                                                     head=node.getNext();
41.                                                     node = null;
42.                                                    }
43.                                                    else
44.                               if(node==tail)
45.                                                    {
46.
node.getPrev().setNext(null);
47.                                                     tail=node.getPrev();
48.                                                     node = null;
49.                                                    }
50.                                                    else
51.                                                    {
52.                              node.getPrev().setNext(node.getNext());
53.
     node.getNext().getNext().setPrev(node.getPrev());
54.                                                     node= null;
55.                                                    }
56.                               flag=true;
57.                               System.out.println("The  Found  Nuber  And
Deleted");
58.                               break;
59.                              }
60.    node=node.getNext();
61.  }
62.
63.  if(!flag)System.out.println("Not Found Nuber");
64.
65.  //print DNODE
66.  node=head;
```

```
67.  while(node!=null)
68.  {
69.   System.out.print(node.getElement()+" ");
70.   node=node.getNext();
71.  }
72.  System.out.println();
73.  System.exit(0);
74. }
75. }
```

شرح المثال:

في السطر 37 نستفسر إذا كانت العقدة هي الرأس فإنها حالة خاصة إي الحذف من البداية وسبق وان تكلمنا عن هذه الحالة وفائدة هذا الشرط هو الحفاظ على مكان الرأس وهو نقلة بمقدار واحد للإمام و بعد ذلك حذف العقدة.
وفي السطر44 نستفسر إذا كانت العقدة هي الذيل فإنها حالة خاصة أيضاً إي الحذف من النهاية وسبق وان تكلمنا عن هذه الحالة وفائدة هذا الشرط هو الحفاظ على مكان الذيل وهو نقلة بمقدار واحد للخلف و بعد ذلك حذف العقدة.
وإلا ستكون العقدة بين الرأس والذيل فإنها حالة خاصة أيضاً , فيتم ربط مؤشر العقدة السابقة مع العقدة التالية وربط مؤشر العقدة التالية مع العقدة السابقة كما في الأسطر (51 - 55) والشكل 24-15 يبين ذلك .

دمج القوائم الثنائية: بعمل هذا المثال على دمج قائمتين ثنائيتين:



لشكل 24-15

```
1. //برنامج دمج قائمتين ثنائيتين
2. class raed_9 extends DNODE {
3. public static void main(String args[]){
4.   DNODE head=null;          // head DNODE of the list
5.   DNODE tail=null;          // tail DNODE of the list
6.   DNODE node=null;
7.
8.   DNODE head2=null;// head2 DNODE of the list
```

```
9.   DNODE tail2=null;              // tail2 DNODE of the list
10.   DNODE node2=null;
11.    int size=5;
12.  //insert DList1
13.  tail=head=node=new DNODE(0);
14.  for (int i=1;i<size;i++)
15.  {
16.   node=new DNODE(i);
17.   tail.setNext(node);
18.   node.setPrev(tail);
19.   tail=node;
20.  }
21.  //insert DList2
22.  tail2=head2=node2=new DNODE(10);
23.  for (int i=11;i<size+10;i++)
24.  {
25.   node2=new DNODE(i);
26.   tail2.setNext(node2);
27.   node2.setPrev(tail2);
28.   tail2=node2;
29.  }
30.
31.    //Print DList1
32.  node=head;
33.  while(node!=null)
34.  {
35.   System.out.print(node.getElement()+" ");
36.   node=node.getNext();
37.  }
38.  System.out.println("\nDList1");
39.
40.  //Print DList2
41.  node2=head2;
42.  while(node2!=null)
43.  {
44.   System.out.print(node2.getElement()+" ");
45.   node2=node2.getNext();
46.  }
47.  System.out.println("\nDList2");
48.
49.  //node+node2
50.  tail.setNext(head2);
51.  head2.setPrev(tail);
52.  tail=tail2;
53.  tail2=head2=node2=null;
54.
```

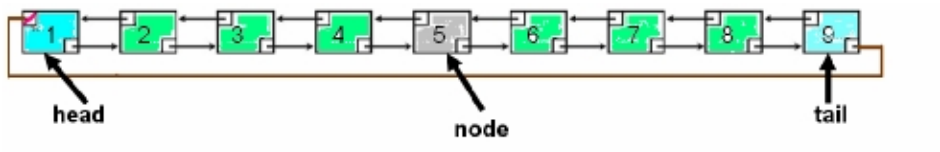
```
55.  //Print DList1 + DList2
56.  node=head;
57.  while(node!=null)
58.  {
59.   System.out.print(node.getElement()+" ");
60.   node=node.getNext();
61.  }
62.  System.out.println("\nDList1 + DList2");
63.
64. }
65. }
```

شرح المثال:
الأسطر (53 - 50) تمت دمج اللائحتين مع بعضها البعض عن طريق جعل ذيل اللائحة الأولى يؤشر إلى رأس اللائحة الثانية كما في السطر 50. و جعل رأس اللائحة الثانية يؤشر إلى ذيل اللائحة الأولى كما في السطر 51 . وأخر عملية هي مساواة مؤشر اللائحة الأولى بمؤشر اللائحة الثانية كما في السطر 52.
السطر 53 يعمل على حذف المؤشرات الزائدة من عملية الدمج.

إلى هنا قد اتضحت عمل القوائم المذبلة وهذه الأمثلة التي كتبت إذا فهمها القارئ فأننا نضمن له أن إي سؤال سيواجه سيعرف إجابته بلا تعب أو مجهود.

مثال مهم :

لنفترض أن لدينا لائحة ثنائية دائرية بداخلها هذه كما في الشكل 25-15



شكل 25-15

ولدينا ثلاثة مؤشرات node , tail , head
وكان أمر الطباعة

```
System.out.println(node.getElement());

System.out.println(node.getNext().getNext().getElement());

System.out.println(node.getNext().getPrev().getElement());

System.out.println(node.getNext().getNext().getNext().getNext().getNext
().getNext().getElement());
System.out.println(head.getElement());

System.out.println(head.getNext().getElement());
```

```
System.out.println(head.getPrev().getPrev().getElement());

System.out.println(tail.getPrev().getPrev().getNext().getElement());

System.out.println(tail.getNext().getPrev().getElement());
```

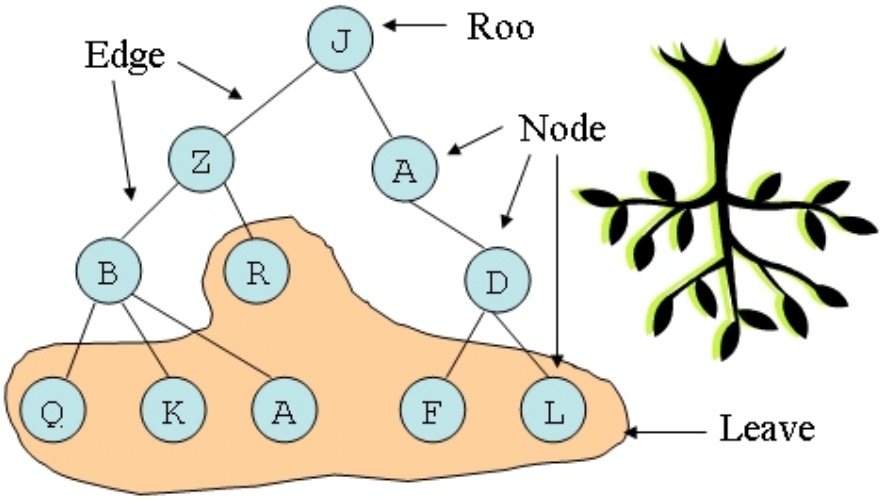
فما هو ناتج تنفيذ البرنامج؟

بإمكاننا أن نقدر ثمن خوارزم معالجة اللوائح بواسطة :

- **§** الحجم المشغول في الذاكرة .
- **§** عدد المؤشرات التي من الواجب عبورها أو استعمالها.

1.7 الأشجار (Trees)

الشجرة هي مكان غير خطي لتخزين المعلومات ، وتستخدم الأشجار لتمثيل المعلومات التي لها علاقات تشعبيه بحيث أن كل عنصر "معلومة" في الشجرة له أب واحد وقد يكون له صفر أو أكثر من الأبناء.
الجذر Root هو العنصر الوحيد الذي لا يوجد له أب كما في الشكل 15-26.



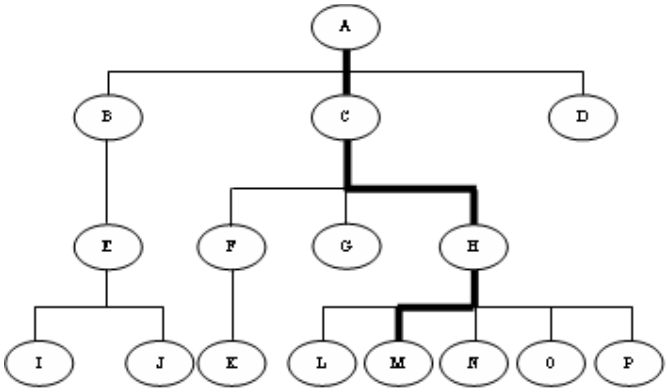
شكل 15-26

1.7.1 مصطلحات الأشجار

عناصر الأشجار تسمى خلايا Nodes ، وكل خلية لها مسار Path واحد فقط يوصلها بالجذر Root.
والمسار هو عبارة عن مجموعة خلايا متتابعة للوصول إلى خلية معينة. طول المسار Path Length هو عبارة عن عدد الوصلات من الجذر إلى الخلية المراد معرفة طول مسارها والذي يساوي عدد الخلايا ناقص واحد.
في الشجرة التالية المسار (M,H,C,A) يوصل الخلية M بالجذر A طوله 3.

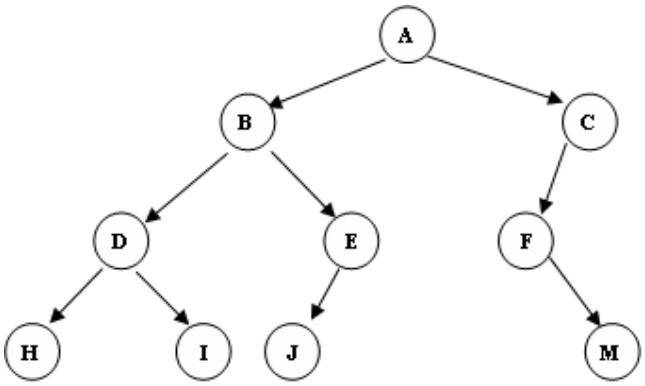
عمق Depth الخلية هو طول مسارها إلى الجذر ، مثلاً الخلية E عمقها 2. الجذر A عمقه 0.
المستوى Level هو كل الخلايا التي لها نفس العمق. المستوى الثاني عبارة عن {E,F,G,H}.
إرتفاع Height الشجرة هو أكبر عمق موجود للشجرة، وفي الشجرة السابقة يساوي 3.
الشجرة التي يوجد بها خلية واحدة فقط إرتفاعها يساوي 0. والشجرة التي لا تحتوي على أي خلية يعرف إرتفاعها (1-).
درجة الخلية Degree هو عدد أبنائها الخلية H درجتها 5.

الورقة Leaf هي الخلية التي درجتها صفر أي لايوجد لها أبناء.
ومن الشكل 27-15 يتضح جميع ما سبق.



شكل 27-15

1.7.2 الأشجار الثنائية (Binary Trees)



شكل 28-15

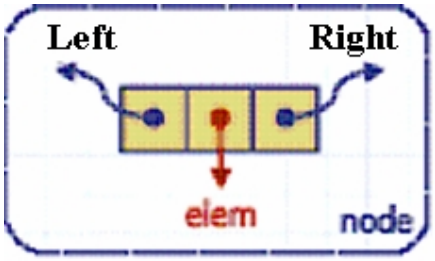
الأشجار الثنائية هي أشجار كسابقتها في التعريف إلا أن عدد الأبناء لأي خلية لا يتجاوز الاثنان

طرق تمثيل الشجرة الثنائية

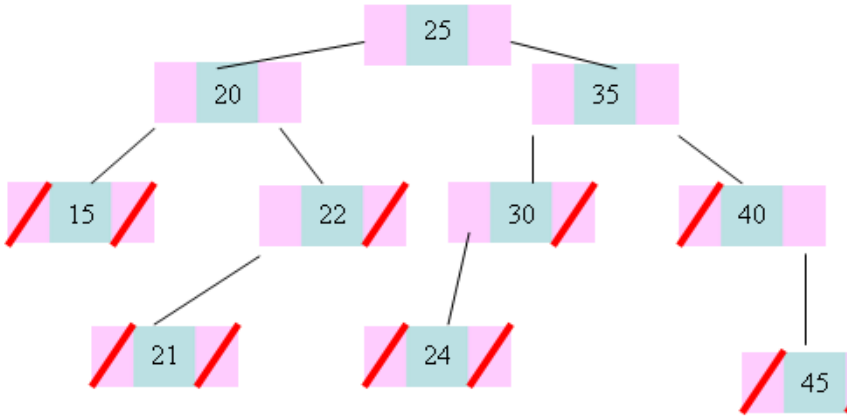
أن الشجرة تمثل بعدة طرق بواسطة القوائم, وتمثل بالمتجهات وسندرس فيما يلي هذه الطرق:

- **v** إذا كانت الشجرة ثنائية منتظمة بعمق (H) فإن عدد عناصر هذه الشجرة يساوي (2^(H+1)-1) وعلية فإنها تمثل بمصفوفة أحادية البعد و عدد عناصر المتجه الذي يمثل الشجرة يساوي (2^(H+1)-1) . ومميزات هذه الطريقة
  - **ý** السهولة فإذا أعطيت موقع العقدة الابن فمن السهل تحديد موقع الأب بالنسبة لها. فلو كانت العقدة الابن في الموقع n من المصفوفة فإن موقع الأب يكون صحيحاً(n/2).
  - **ý** تطبق بسهولة في لغات البرمجة , مثل بيسك وفورتران حيث تكون مواقع الذاكرة الثابتة متوفرة مباشرة .

  وعيوب هذه الطريقة
  - **ý** عملية الإضافة والحذف تؤدي إلى تحريك البيانات إلى أعلى وأسفل في المصفوفة وهذا يضيع وقت المعالجة .
  - **ý** تكون هنالك مواقع ذاكرة غير مستغلة
- **v** **تمثيل الشجرة بواسطة القوائم :**



شكل 15-29



شكل 15-30

ونلاحظ من الشكل السابق أن كلما اتجهنا إلى اليمين يكون العدد اكبر من السابق وكل ما اتجهنا إلى اليسار كان العدد اصغر من الأب أو العقدة السابقة وهكذا تكون الأعداد مرتبة تصاعدياً وتنازلياً .

ونكرر أن الأعداد لا تتكرر في الشجرة الثنائية.

ونلاحظ أن من عيوب هذه الطريقة

- تحتوي على فراغ في فضاء الذاكرة غير مستعمل نتيجة لاستخدام مؤشر صفرية NULL.
- خوارزمية التطبيق لها أصعب اللغات في اللغات التي لا تعطي تقنية ذاكرة متحركة (ديناميكية) .

1.7.3 تطبيقات الأشجار الثنائية

i. شجرة Huffman لضغط البيانات

لشرح كيفية عمل طريقة Huffman نفرض انه يوجد لدينا ملف وحجمه byte1000 يحتوي على الحروف (a,b,c,d,e,f,g,i,j) .

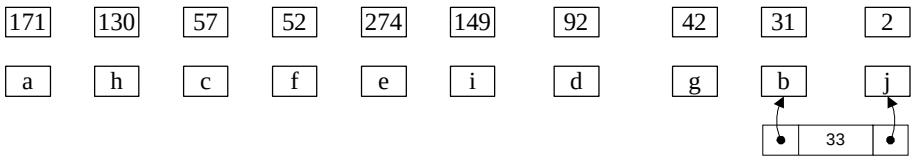
1- نقوم بعمل إحصائية عن الملف المراد تقليص حجمه وذلك بعدّ تكرار كل حرف.

جدول 15-2

a	b	c	d	e	f	g	h	i	J
171	31	57	92	274	52	42	130	149	2

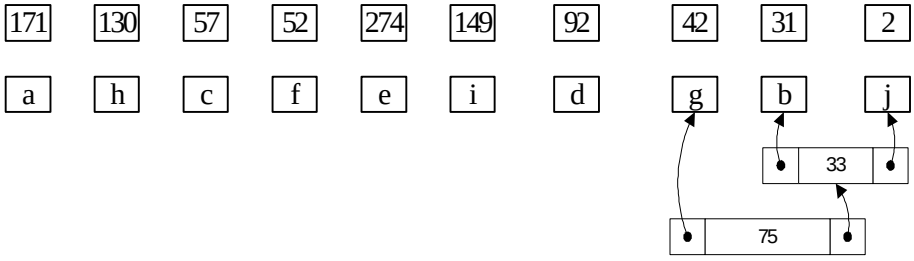
2- نقوم ببناء Binary Tree وذلك عن طريق اختيار الحروف ذات الأقل تكرار وتوصيلها ببعض.

في هذا المثال حرف j وحرف b يمثلان أقل تكرار ، نقوم بتوصيلهما مع وضع مجموع تكرارهما كما في شكل 31-15.



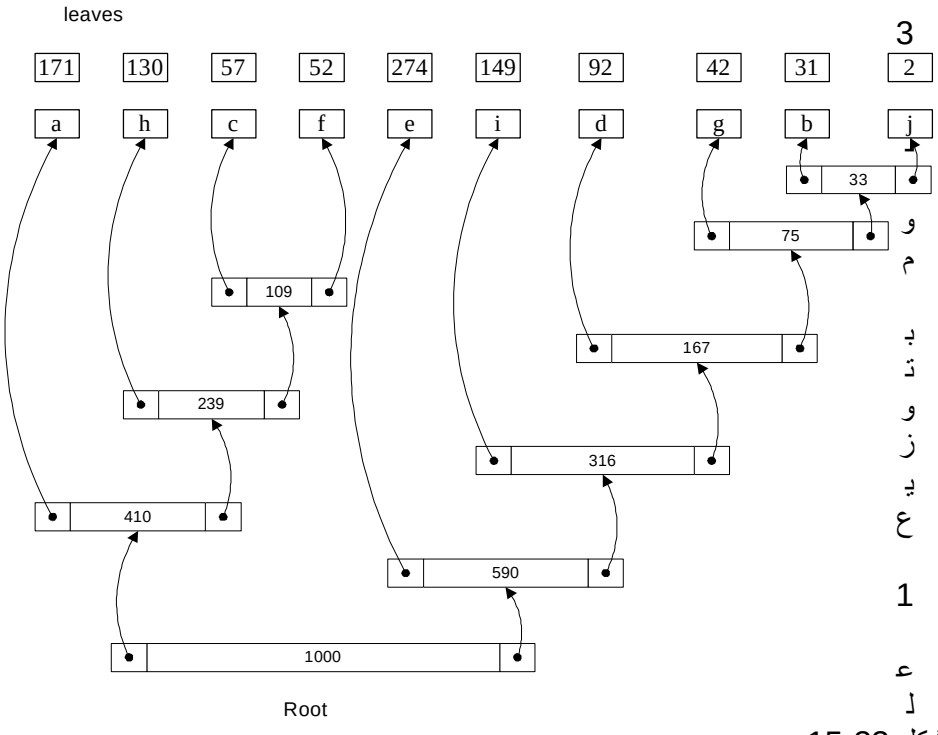
شكل 31-15

مثل مجموع حرفي b,j وحرف g الأقل تكرارا نقوم بتوصيلهما بنفس الخطوات السابقة



شكل 29-15

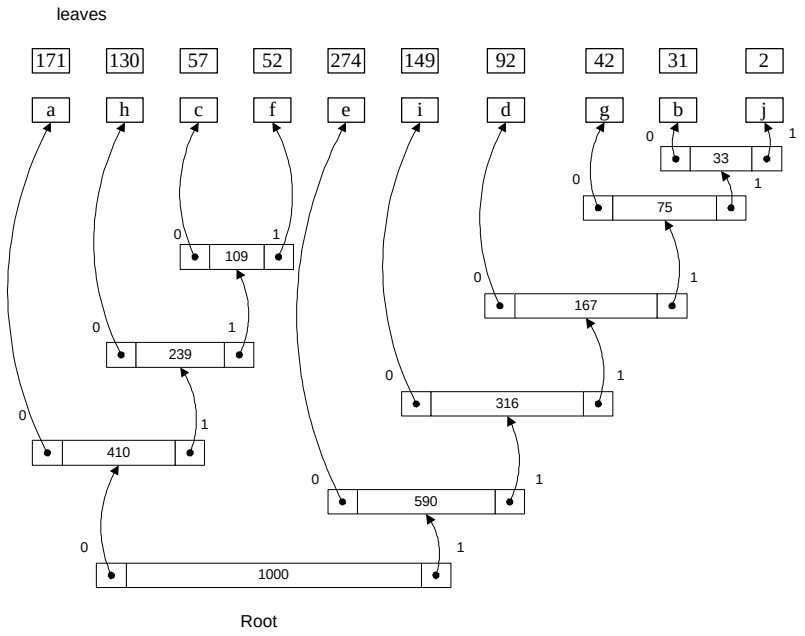
وصيل جميع الحروف تكون Binary Tree كالشكل 32-15:



شكل 33-15

كل فرع من الفروع اليمنى ، و 0 على الفروع اليسرى.

4- نقوم بتسجيل المسارات التي توصل إلى كل حرف من الحروف الأصلية للملف ، وذلك بتتبع مسارات Binary Tree من الجذر Root إلى الأوراق Leaves.



شكل 33-15

جدول 3-15

a	b	c	d	e	f	g	h	i	J
00	111110	0110	1110	10	0111	11110	010	110	111111

عند تكوين الملف المضغوط نستبدل الحروف الأصلية بمساراتها المحسوبة في الخطوة رقم 4.

ونستطيع حساب حجم الملف الجديد ونسبة تقليصه بضرب طول مسار كل حرف في تكراره في الملف

جدول 4-15

a	b	c	d	e	f	g	h	i	j	
00	111110	0110	1110	10	0111	11110	010	110	111111	
2	7	4	4	2	4	5	3	3	6	
171	31	57	92	274	52	42	130	149	2	
342	217	228	368	548	208	210	390	447	12	2970

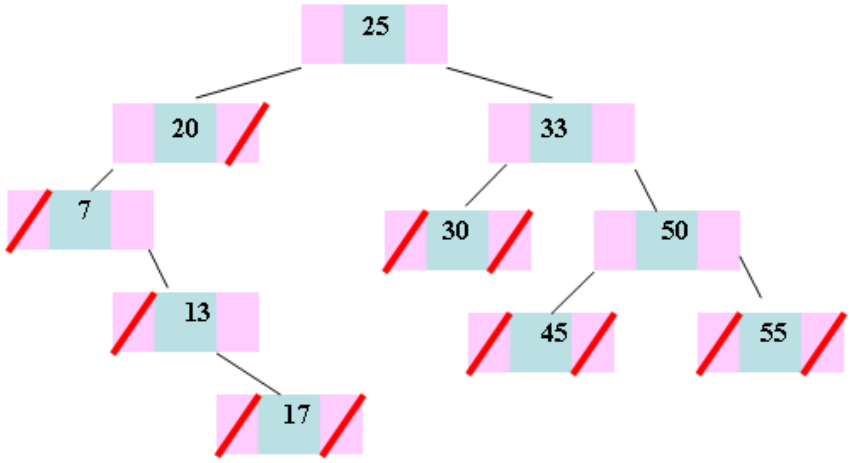
أي أن الملف الجديد سيكون حجمه 2970/8= 372 بايت ، بمعنى أن الحجم الجديد يعادل 37.2% من الحجم الأصلي.
فكرة عمل هذه الطريقة تتلخص في إنها تستغل الحروف الموجودة بكثرة في الملف وتضعها في أقصر مسار في Binary Tree مما يعني إنها (الحروف) سيتم أعطائها أقل حيز تخزيني ممكن.

1.7.4 خوارزمية بناء الشجرة الثنائية

أن بناء الشجرة الثنائية يعتمد على طريقة عبورها, وسابقاً قد سقنا عدة طرق لعبور الشجرة بالاعتماد على المؤشرات وفي الخوارزميات التالية جميعها سنفترض الشجرة الثنائية ذات مؤشر الإباء بالأبناء (Father Link) . وعملية إضافة عقدة في الشجرة الثنائية تعتمد على قيمة تلك العقدة , فإن كانت القيمة اكبر من الجذر اتجهنا إلى اليمين والعكس نتجه إلى اليسار وخلاصة هذه الخوارزمية تتلخص بالأتي:

- ý ضع العنصر الأول على أساس أنة العقدة الأول في الهيكل (الجذر) .
- ý والعدد التالي إذا كان العنصر المراد إدخاله اكبر من الجذر سنضعه على يمين الجذر .
- ý وإلا على يسار الجذر .

والشكل 34-15 يبين ذلك . فإن أدخلنا هذه القيم (25,20,7,13,33,50,45,17,30,55) ستكون الشجرة بهذا الشكل



شكل 34-15

لقد بدئنا بالجدر (25) ثم انتقلنا إلى اليسار بالعدد 20 لأنة اصغر من 25 ثم انتقلنا إلى يسار 25 و 20 بالعدد 7 لأنة أصغر من 20 ثم انتقلنا بالعدد 13 إلى اليسار من 25 وهكذا إلى نهاية الأعداد .

ونعرض الآن صنف الأشجار الثنائية :

```
1. /**
2.  * Class binary tree by storing references to
3.  * an element, a parent node, a left node, and a right node.
4.  */
5. public class BTNode{
6.   private int element;                    // element stored at
this node
7.   private BTNode left, right;  // adjacent nodes
8.   /** Main constructor */
9.   public BTNode(){}
10.
11.   public BTNode(int element) {
```

```
12.     setElement(element);
13.     setLeft(null);
14.     setRight(null);
15.                                                 }
16.   /** Returns the element stored at this position */
17.   public int element() { return element; }
18.   /** Sets the element stored at this position */
19.   public void setElement(int o) { element=o; }
20.   /** Returns the left child of this position */
21.   public BTNode getLeft() { return left; }
22.   /** Sets the left child of this position */
23.   public void setLeft(BTNode v) { left=v; }
24.   /** Returns the right child of this position */
25.   public BTNode getRight() { return right; }
26.   /** Sets the right child of this position */
27.   public void setRight(BTNode v) { right=v; }
28. }
```

وفيما يلي البرنامج الذي ينفذ جميع ما سبق :

```
1. //برنامج الشجرة الثنائية
2. class raed_10 extends BTNode {
3. public static void main(String args[]){
4.   BTNode root=null;                    // root BTNode of the
BTree
5.   BTNode right=null;            // right BTNode of the BTree
6.   BTNode left=null;                    // left BTNode of the
BTree
7.   BTNode node=null;
8.
9.    int Arr[]={5,6,2,8,4,10,18,9,0};
10.  //insert BTree
11.  for (int i=0;i<Arr.length;i++)
12.  {
13.   if(root==null)
14.                    {
15.                         root=node=new BTNode(Arr[0]);
16.                    }
17.                    else
18.                    {
19.                         node=new BTNode(Arr[i]);
20.                         BTNode s,p;
21.                         p=s=root;
22.                         while(s!=null)
23.                                        {
24.                                          p=s;
```

```
25.
if(node.element()>s.element())
26.                                         s=s.getRight();
27.                                         else
28.                                         s=s.getLeft();
29.                                     }
30.                         if(node.element()>p.element())
31.                                         p.setRight(node);
32.                                         else
33.                                         p.setLeft(node);
34.
35.                     }
36.  }
37.
38.    //Print DBTree
39.  Print(root);
40.  System.out.println("\nDBTree");
41. }
42. static void Print(BTNode node){
43.  if(node!=null){
44.                         System.out.print(node.element()+" ");
45.                         Print(node.getLeft());
46.                         Print(node.getRight());
47.                     }
48.                                     }
49. }
```

شرح المثال:
الأسطر (18 - 35) قمنا بعمليه البحث عن الموقع التي سنضع العقدة حسب خوارزمية الأشجار فإذا كانت القيمة اكبر اتجهنا يساراً وإلا اتجهنا يميناً إلى أن يصل s = null كما في السطر 22.

فائدة المتحول p هو عبارة مؤشر يؤشر بمقدار واحد للخلف لنحتفظ بموقع أخر عقدة الناتجة من عملية البحث .

بعد من بحث الموقع نقوم بعمليه استفسار فإذا كانت القيمة اكبر من العقدة التي عثرنا عليها بواسطة p فأننا نضع العقدة على اليمين وإلا نضعها على اليسار وهكذا لباقي العقد كما في الأسطر (34 – 30) .

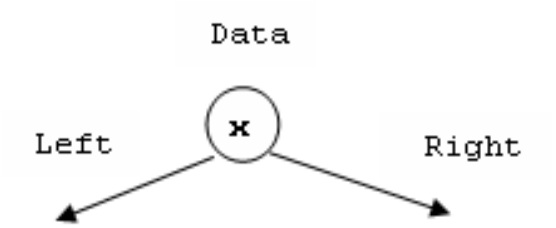
السطر 39 تم استدعاء منهج Print ليقوم بطباعة الشجرة . و هنا تكون عملية الطباعة بالاستدعاء الذاتي فهي أسهل .

1.7.5 خوارزم إسترجاع المعلومات من الأشجار الثنائية

Binary Tree Traversal

والمقصود هنا هو زيارة كل خلية واسترجاع معلوماتها "للطباعة مثلاً" مرة واحد فقط. ولنفرض أن تمثيل الشجرة الثنائية كما في الشكل 35-15 :

فمن الشكل السابق يمكننا استرجاع البيانات
من الشجرة الثنائية بست طرق وهي :



شكل 15-35

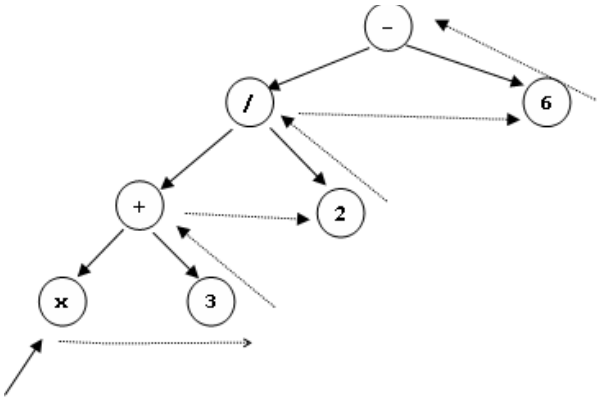
Left Right Data
Left Data Right
Right Left Data
Right Data Left
Data Left Right
Data Right Left

وباستبعاد الطرق التي توجد بها Right قبل Left يتبقى لنا ثلاث طرق وهي :

الأولى وتسمى Post Order وتمثل Left Right Data
الثانية وتسمى Pre Order وتمثل Data Left Right
والثالثة وتمسى In Order وتمثل Left Data Right
وسميت هذه الطرق نسبة إلى موقع الـ Data ، ففي الطريقة الأولى تقع Data الأخيرة ...وهكذا..

1. طريقة Post Order .

```
static void PostOrder(Node ptr)
{
   if(ptr!=null){
     PostOrder(ptr.getLeft());
     PostOrder(ptr.getRight());
     System.out.print(ptr.element()+" ");
          }
}
```



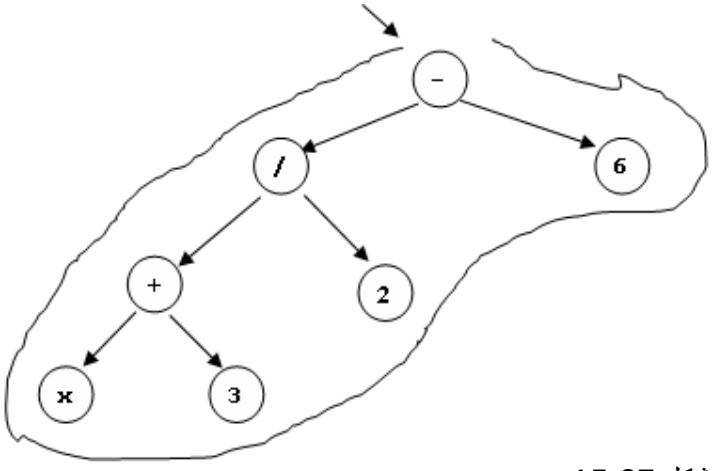
بالنظر إلى عمل هذه الطريقة نجدها تسترجع البيانات من المستويات الأعمق أولاً ، ثم التي تليها وهكذا، مع أولوية الجهة اليسرى للشجرة.

شكل 15-36

2. طريقة Pre Order

```
static void PreOrder(Node ptr)
{
   if(ptr!=null){
     System.out.print(ptr.element()+" ");
     PreOrder(ptr.getLeft());
     PreOrder(ptr.getRight());
          }
}
```

يمكن إسترجاع البيانات بالرسم وذلك برسم محيط حول الشجرة بادية من الجذر ، مع أولوية الجهة اليسرى للشجرة.

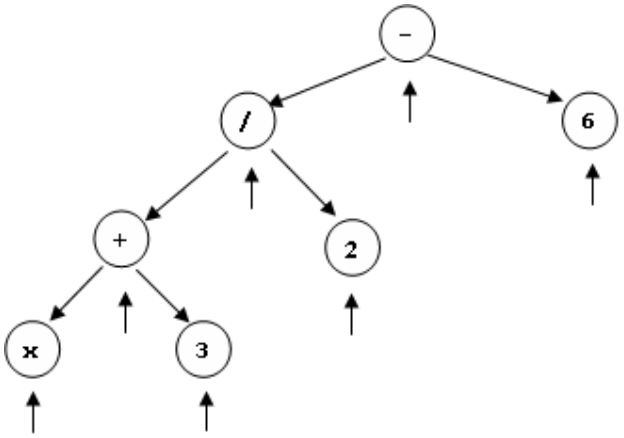


شكل 37-15

b. طريقة In Order

```
static void InOrder (Node ptr)
{
  if(ptr!=null){
    InOrder (ptr.getLeft());
    InOrder (ptr.getRight());
    System.out.print(ptr.element()+" ");
          }
}
```

يمكن إسترجاع البيانات من الرسم وذلك بإسترجاع البيانات الموجودة في أقصى اليسار ، بغض النظر عن المستوى ... وهكذا.

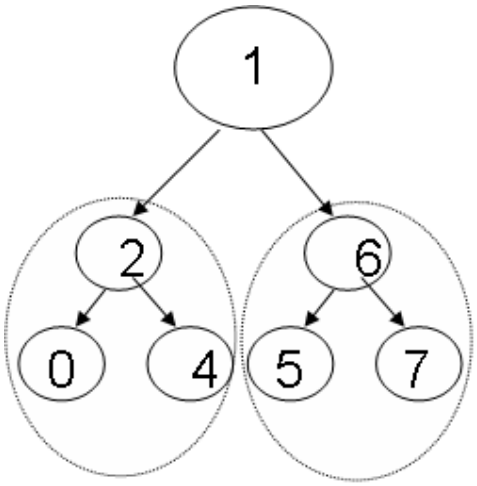


شكل 38-15

1.7.6 خوارزمية عد عقد الأشجار :

إن حجم الشجرة يساوي عدد العقد في الشجرة الفرعية اليمنى مضافاً إليها عدد العقد في الشجرة الفرعية اليسرى مضافاً إليها عقدة الجذر.

طباعة أب وأخ العدد المدخل كما في الشكل 39-15



شكل 39-15

فلو أدخلنا الرقم 6 فأنة سوف يطبع لنا الأب وهو 1 و يطبع لنا الأخ وهو 2 .
وهذا كود البرنامج:

```
1. //برنامج يطبع أخ و أب عدد مدخل بواسطة الشجرة الثنائية
2. import javax.swing.JOptionPane;
3. class raed_11 extends BTNode {
4. public static void main(String args[]){
5.  BTNode root=null;                    // root BTNode of the
BTree
6.  BTNode right=null;                      // right BTNode of the
BTree
7.  BTNode left=null;                    // left BTNode of the
BTree
8.  BTNode node=null;
9.
10.   int Arr[]={5,6,2,8,4,10,18,9,0};
11.   //insert BTree
12.   for (int i=0;i<Arr.length;i++)
13.   {
14.    if(root==null)
15.                          {
16.                               root=node=new BTNode(Arr[0]);
17.                          }
18.                          else
19.                          {
```

```
20.                                 node=new BTNode(Arr[i]);
21.                                 BTNode s,p;
22.                                 p=s=root;
23.                                 while(s!=null)
24.                                                 {
25.                                                  p=s;
26.
if(node.element()>s.element())
27.                                                 s=s.getRight();
28.                                                 else
29.                                                 s=s.getLeft();
30.                                                }
31.                                 if(node.element()>p.element())
32.
        p.setRight(node);
33.                                                             else
34.
        p.setLeft(node);
35.
36.                             }
37.   }
38.
39.   //Print DBTree
40.   Print(root);
41.   System.out.println("\nDBTree");
42.
43.   String snum1;
44.   int num1;boolean flag=false;
45.   snum1 = JOptionPane.showInputDialog("Enter Number:")
;
46.   num1 = Integer.parseInt(snum1);
47.
48.   if(root.element()==num1)
49.                                 {
50.                                  System.out.println
51.                                  ("The Number Not Prather And
Father");
52.                                  System.exit(0);
53.                                 }
54.   //Sertch The Number In Tree
55.   BTNode p=null;
56.   node=root;
57.   boolean f=true;
58.   while(f)
59.                   {
60.                    p=node;
```

```
61.                     if(node.element()>num1)
62.                         node=node.getLeft();
63.                         else
64.                         node=node.getRight();
65.
66.                     if(node==null)break;
67.
68.                     if(node.element()==num1){
69.                                             f=false;
70.                                             break;
71.                                         }
72.                 }
73.   if(f)
74.           {
75.                 System.out.println("Not Found Number");
76.                 System.exit(0);
77.           }
78.
79.   System.out.println(p.element()+" Father");
80.
81.   if(p.element()<num1)
82.         {
83.         if(p.getLeft()!=null)
84.
      System.out.println(p.getLeft().element()+" Prather");
85.                         else
86.                         System.out.println("The  Number  Not
Prather");
87.         }
88.         else
89.         {
90.         if(p.getRight()!=null)
91.
      System.out.println(p.getRight().element()+" Prather");
92.                         else
93.                         System.out.println("The  Number  Not
Prather");
94.         }
95.   System.exit(0);
96.
97. }
98. static void Print(BTNode node){
99.   if(node!=null){
100.                 System.out.print(node.element()+" ");
101.                 Print(node.getLeft());
102.                 Print(node.getRight());
```

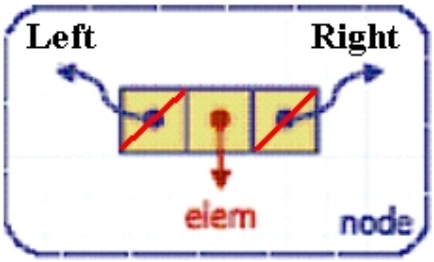
```
103.            }
104.                    }
105. }
```

1.7.7 خوارزمية حساب عدد أوراق الشجرة الثنائية

إن الورقة في الشجرة هي العقدة التي ليس لها أبناء كما قلنا سابقاً , ولحساب عدد أوراق الشجرة , لابد أولاً من التحقق من أن العقدة هي ورقة أم لا وذلك بهذا الشرط

```
if(node.getLeft()==null&&node.getRight()==null)
```

فيكون شكل الورقة كما في الشكل 40-15.



شكل 40-15

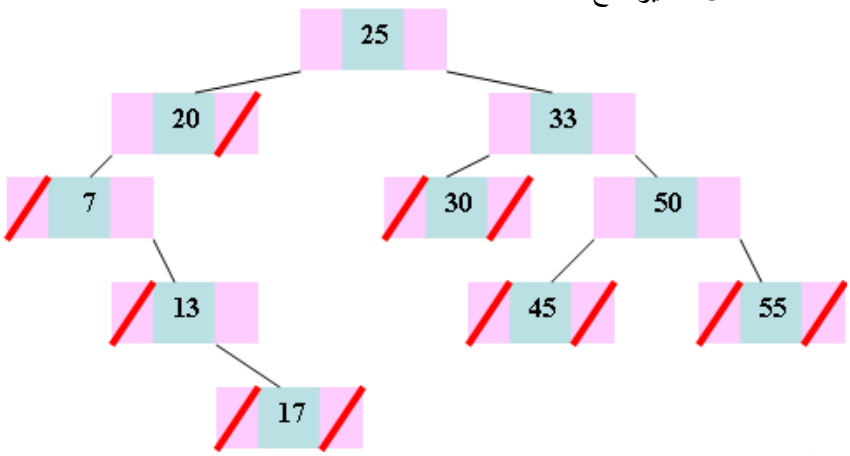
1.7.8 خوارزمية حذف عقد أوراق الشجرة الثنائية

- إذا كان العدد المراد حذفه ورقة لا يحتوي على أبناء فالأمر سهل ولا توجد مشقة

1. نبحث عن العدد وقبل التنقل إلى اليمين أو إلى الشمال نخزن موقعنا في متغير ثم ننتقل وهذا المتغير سيكون يمشي ورآنا بمقدار واحد للخلف إي يكون أب الموقع الحالي .
2. إثناء عمليه التنقل نستفسر عن نوع العقدة فإذا كانت ورقة أم لا بهذا الشرط :

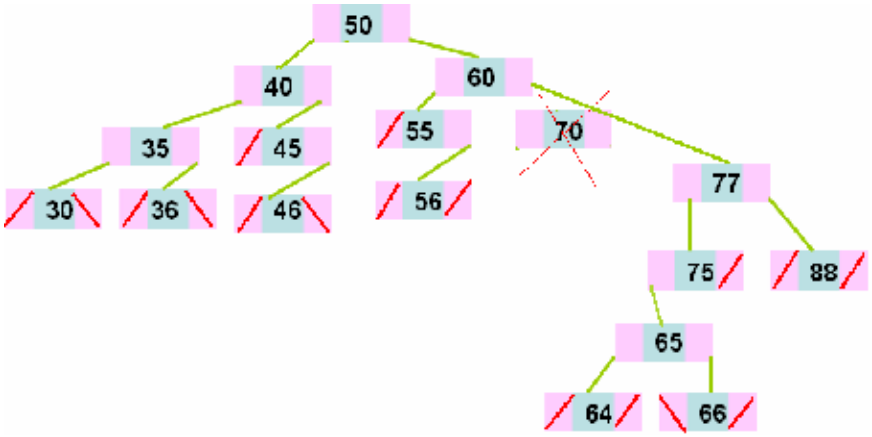
```
if(node.getLeft()==null&&node.getRight()==null)
```

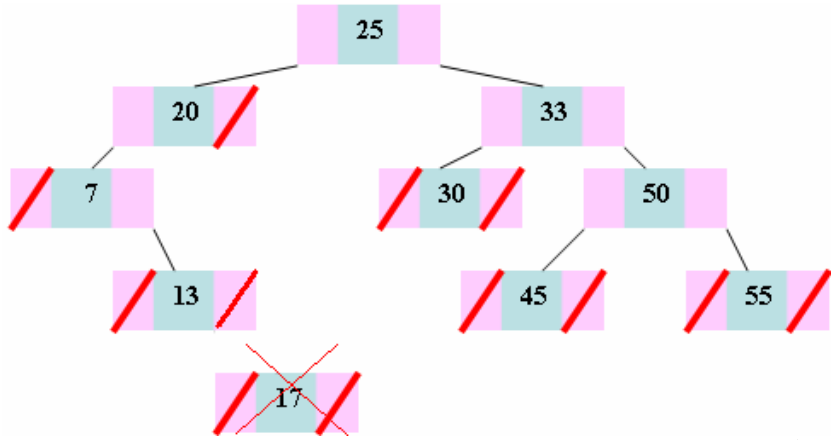
بعد العثور على العدد بقي علينا أن نتأكد هل هو على يسار الأب أم على يمينه فأن كان على يساره نجعل حقل left=NULL فنكون عزلنا العقدة من الشجرة ثم نحذف العقدة ومن الشكل 41-15 سيوضح ذلك .



شكل 41-15

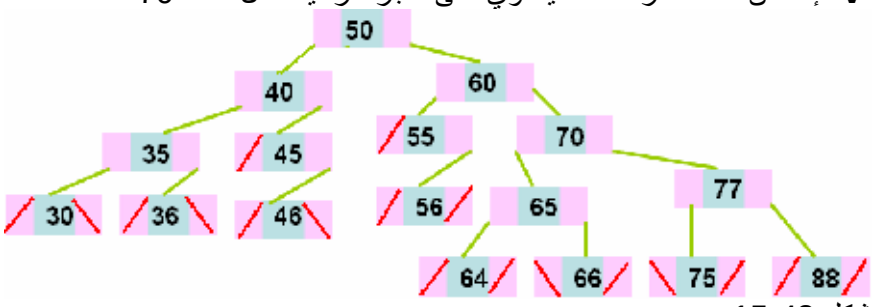
نريد حذف العقدة ذات القيمة 17 فستصبح الشجرة بهذا الشكل





شكل 15-42

- إذا كان العدد المراد حذفه يحتوي على شجرة فرعية مثل العدد 70



شكل 15-43

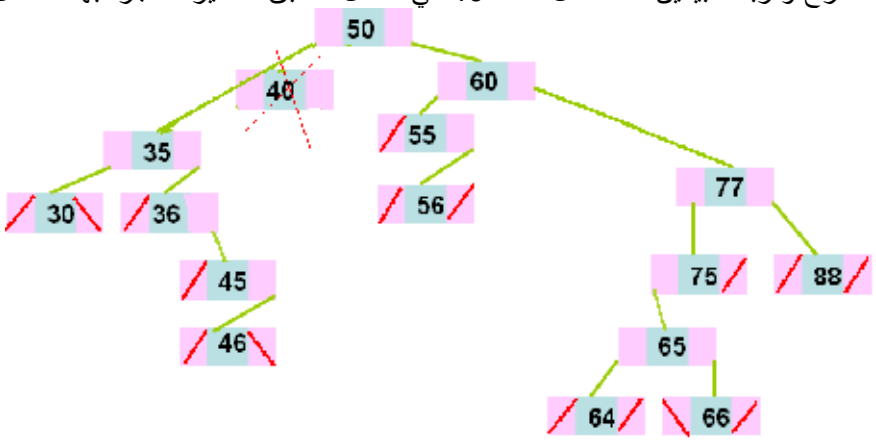
1. نحدد موقع العدد هل هو على يسار الأب أم على يمينه if(60>70).
2. هل العدد يوجد على يمينه فرضاً أعداد وشجرة فرعية if(node->right!=NULL) فإذا تحقق الشرط فأننا نربط يمين الأب بيمين الابن فنكون في هذه الحالة عزلنا 70 بقي علينا ربط
أجزاء الابن وهو نصل لأصغر قيمة في العقدة 77 ونربط يسارها بالعقد التي كانت متصلة بالعدد المراد حذفه فتصير الشجرة بهذا الشكل .

3شكل 15-44

- تم نحذف العقدة .

هذا إن وجدت على يمين العدد عقد وان لم توجد نربط يمين الأب بيسار الابن مباشرة

- v إذا كان العدد اصغر من الأب نفس الخطوات السابقة .
- v إذا كان العدد يوجد بيساره تفرع تربط يسار الأب بيسار الابن ونصل لأكبر عدد داخل التفرع ونربطه بيمين العدد مثل العدد 40 في الشكل السابق فتصير الشجرة بهذا الشكل



شكل 15-45

وإلا نربط يسار الأب بيمين الابن , ثم نحذف العقدة .

- v بقي علينا حالة وهي أن أراد حذف الجذر هي بعض الشيء مربكة إلا إنها بسيطة
  - ý نبحث أولا عن اصغر عقدة في الجذع الأيمن للجذر ونطبق جميع الشروط التي ذكرناها سابقاً .
  - ý ونأخذ قيمة العقدة ونساوي قيمة الجذر بها ثم نحذف العقدة التي بحثنا عنها وبهذا نكون أحللنا قيمة الجذر إي بمثابة حذفنا الجذر وهذا الشكل سينتج .

شكل 15-46

- ý وإن لم يوجد تفرع يمين للجذر فإننا ننقل الجذر بمقدار واحد لليسار ونحذف العقدة .

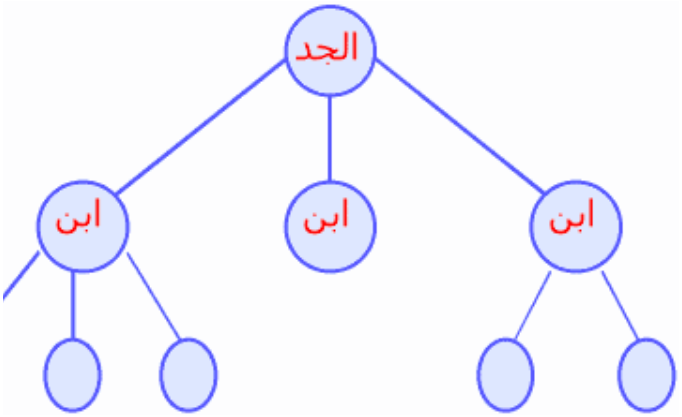
وهذه كل الخطوات يمكن دمجها ببرنامج شامل يستطيع أن يحذف من إي مكان .

إلى هنا ينتهي حديثنا عن الأشجار .

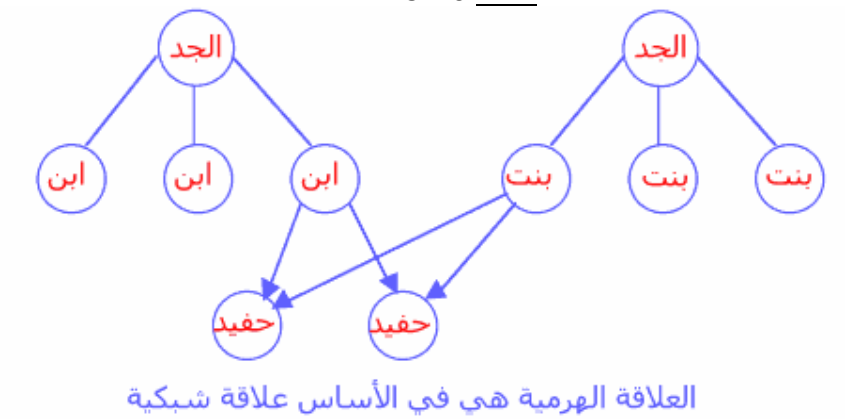
1.8 هياكل البيانات الشبكية (المترابطة)

وتعنى Plex الشبكات (الرسوم Graphs) كما تسمى بيانات مضفرة.إذا اتصل أي عنصر بيان في المستوى الأدنى من هياكل البيانات الشجرية بأكثر من عنصر في مستوى أعلى فيطلق عليه اسم هياكل بيانات شبكية، حتى

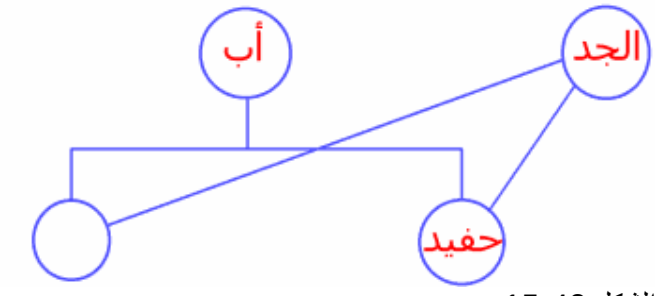
شجرة العائلة من النوع الشبكي وليست من النوع الهرمي فالشكل15-46 يوضح العلاقات الهرمية لأسرة تتكون من جد، أبناء، أحفاد، شكل غير حقيقي لأننا أهملنا الأمهات منذ زمن بعيد لكن الشكل 15-47 وهي علاقة توضحيها أكثر كما في شكل 15-48 و الشكل 15-49 يبين التكافئ الشجري لمثال ما.



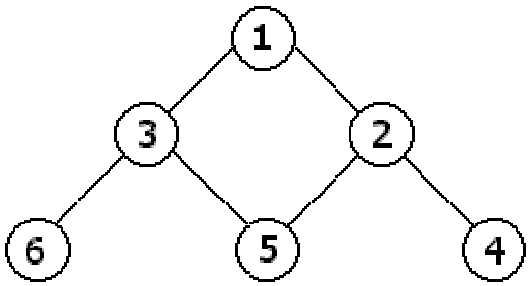
الشكل 15-46



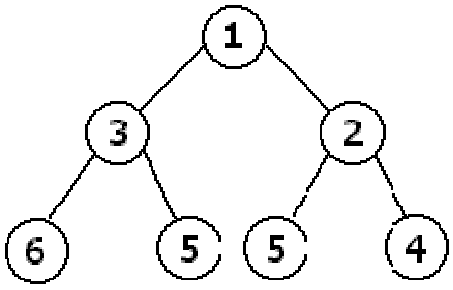
الشكل 15-47



الشكل 15-48

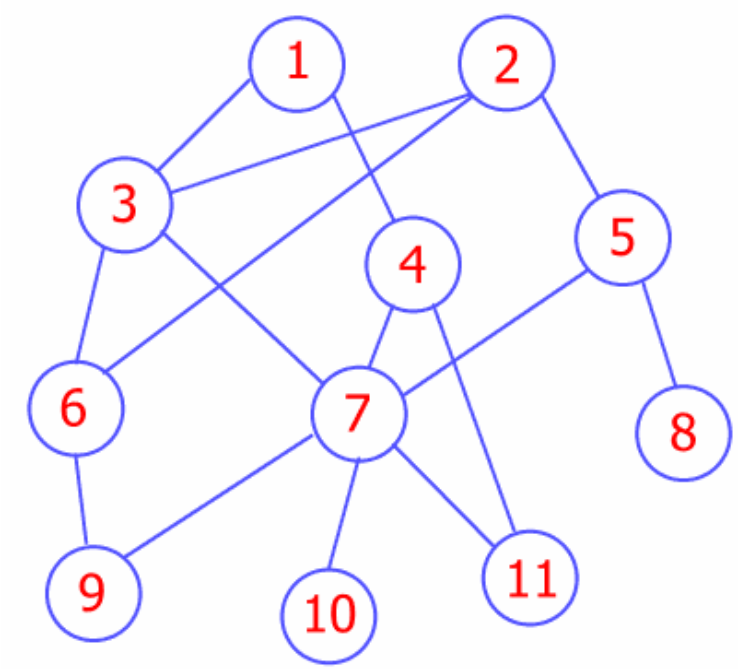


تكافئ الشكل الشجري التالي



الشكل 49-15

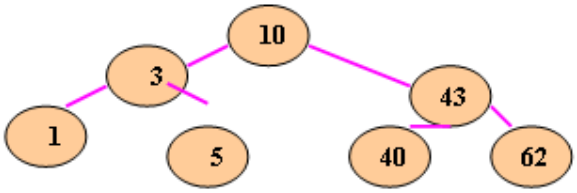
وتنقسم هياكل البيانات الشجرية إلى نوعين، بسيط ومعقد، ففي النوع البسيط يمكن تحديد مستويات الهيكل البنائي للبيانات أما النوع المعقد فيصعب ذلك كثيراً كما في الشكل 50-15 .



الشكل 50-15

تمارين الفصل

1. اكتب برنامج يقوم باستخراج الأعداد المتكررة من المكدس إي بحذف الأعداد المتكررة ؟
2. اكتب برنامج يقوم بعملية ترتيب مكدس ؟
3. اكتب برنامج يقوم بعملية إزاحة يمين وشمال للأعداد حسب الإزاحة المطلوبة من المستخدم؟
4. اكتب برنامج يقوم بدمج مكدسين ويعمل جميع العمليات المنطقية (تقاطع, اتحاد, طرح) ؟
5. اكتب برنامج يقوم بتحويل الأعداد الزوجية بمكدس والفردية بمكدس أخر ؟
6. اكتب برنامج يقوم بحذف الأعداد الأولية من المكدس ؟
7. اكتب برنامج يقوم بعكس طابور ؟
8. اكتب برنامج يقوم بفصل طابور من مكان محدد من قبل المستخدم ؟
9. اكتب برنامج يقوم بحذف المواقع الزوجية بالطابور ؟
10. اكتب برنامج يقوم بتدوير طابور بالاتجاهين يمن وشمال ؟
11. اكتب برنامج يقوم بنسخ مكدس إلى طابور ؟// ملاحظة يوجد فرق بين النسخ والقص//
12. لديك طابور بة أشخاص وهم في المستشفى ومرقمين من 1 إلى 5 وهم في صف الآنتظار فجأة الشخص رقم 3 أصيب بوجع قوي ولا يستحمل الآنتظار فأمر الطبيب بإدخاله لأنة حالة طارئة فكيف تعالج هذه المشكلة بتطبيق إلية وخوارزمية الطابور الأتي أولاً للطبيب الداخل أولاً ؟
13. اكتب برنامج يقوم بترتيب قائمة أحادية ؟
14. اكتب برنامج يقوم بقلب أي عكس قائمة أحادية ؟
15. اكتب برنامج يقوم بقلب المواقع الزوجية مع الفردية أي swap ؟
16. اكتب برنامج يقوم بحذف المواقع الزوجية بالقائمة الأحادية ؟
17. اكتب برنامج يقوم بدمج قائمتين أحاديتين وعمل جميع العلاقات الرياضية (تقاطع , اتحاد , طرح) ؟
18. اكتب برنامج يقوم بطباعة ثالث اكبر قيمة من القائمة ؟
19. اكتب برنامج يقوم بترتيب طابور بواسطة القائمة الأحادية ؟
20. اكتب برنامج بواسطة الدوال يقوم بطباعة ثالث مجموع اكبر القيم وطباعة الأعداد بالقائمة الأحادية مثل
21. (1,2,7,9,5,6) فيطبع 21 والأعداد 7,9,5 ؟
22. اكتب برنامج يقوم بتمثيل بيانات القائمة ألأحادية في الذاكرة بهذا الشكل ؟
23. هذا السؤال للقارئ النبيل ----- بواسطة القائمة الأحادية اكتب برنامج يعمل عمل القائمة الثنائية إي يكون الهيكل العام لقائمه مكون من متغيرين فقط إي next , int لا يوجد مؤشر يؤشر للخلف مثل last فيستطيع العودة للخلف مرة ومرتين ..... الخ ؟
24. بواسطة القوائم ألثنائيه اكتب برنامج يقوم حذف المواقع الأولية من هذه القائمة ومعروف إن الأعداد الأولية (1,2,3,5,7,9,11,13,.....) ؟
25. بواسطة ألعوديه أو التكرار اكتب برنامج يعمل على طباعة عدد العقد الثنائية ؟
26. جميع الأسئلة التي ذكرت في الفصول السابقة ينبغي على القارئ تطبيقها بالقوائم الثنائية ؟
27. اكتب برنامج يقوم بتمثيل بيانات هذا الشكل في الذاكرة عم مراعاة نوع القوائم ؟
28. اكتب برنامج بواسطة الأشجار الثنائية يقوم بطباعة الشجرة بدون استخدام الاستدعاء الذاتي ؟
29. اكتب برنامج بواسطة الأشجار الثنائية يطبع مجموع ما تحت العدد المدخل وكم أعداد اكبر منة وكم أعداد اصغر منة ؟
30. اكتب برنامج بواسطة الأشجار الثنائية يقوم بطباعة الشجرة مرتبة تصاعدي ومرة تنازلي ؟
31. اكتب برنامج بواسطة الأشجار الثنائية يطبع كم عدد الآباء الذين لديهم أبناء اثنان وكم الذين لديهم ابن واحد وكم الذين ليس لهم أبناء ؟
32. ليكن لديك البيانات التالية (10,3,43,62, 5,1,40)



33. المطلوب منك طباعه هذه الشجرة أو إي شجرة بالشكل الأتي

```
            62
        43
            40
10
            5
        3
            1
```

34. اكتب برنامج يقوم بتحويل شجرة ثنائية إلى طابور بشرط أن تدخل البيانات مرتبة للطابور وبدن استخدام إي خوارزمية ترتيب ؟
35. اكتب برنامج يقوم بتحويل طابور إلى شجرة ثنائية علماً أن البيانات في الطابور متكررة والشجرة الثنائية لا تقبل القيم المتكررة ؟
36. لديك الكلمات التالية (SAMI , AMMAR , AHMED , BASSAM , SANAD , MOSTAFA , READ , ALI , KAMAL ,AMIN) المطلوب تكوين شجرة ثنائية مثل المثال الذي تكلمنا علية سابقاً ؟
37. اكتب برنامج يقوم بحذف الأعداد الأولية من الشجرة الثنائية ؟