خوارزميات الفرز الداخلي
Sorting Algorithms

نظراً إلى العدد الكبير لخوارزميات الفرز الداخلي، فإننا سنصنف هذه الخوارزميات ضمن فئات تتفق فيما بينها بالمبدأ العام، وسنعطي مثالاً أو مثالين على كل فئة.

1. الفرز بالاختيار

مبدأ الفرز بالاختيار هو البحث عن أصغر عنصر في السلسلة، ووضعه في أول موقع من السلسلة والمتابعة مع بقية عناصر السلسلة.

مثال

يبين الشكل التالي الخطوات المتتالية للفرز بالاختيار لسلسلة تحوي ستة عناصر.

Data	130	120	30	62	40	20
Selection						20
Data	20	130	120	30	62	40
Selection						30
Data	20	30	130	120	62	40
Selection						40
Data	20	30	40	130	120	62
Selection						62
Data	20	30	40	62	130	120
Selection						120
Data	20	30	40	62	120	130

في كل مرحلة نقوم بالبحث عن أصغر عنصر، ونبدله بأول عنصر في السلسلة المراد فرزها.

يمكن التعبير عن هذه الطريقة في الفرز بواسطة الإجرائية العودية التالية:

```
public static void Select_Sort(int[] t, int i) {
    int j, k, w;
    int n = t.length;
    {
      if (i < n) {
        j = i;
        for (k = i + 1; k < n; k++)
          if (t[k] < t[j])
            j = k;
        w = t[j];
        t[j] = t[i];
        t[i] = w;
        Select_Sort(t, i + 1);
      }
    }
    return;
  }
```

يمكن تحويل هذه الإجرائية العودية إلى إجرائية تكرارية باتباع القواعد المعروفة لهذا التحويل:

```
public static void Itr_Select_Sort(int[] t)
  {
  int j, k, w,i;
 int n = t.length;
 {
       i = 0;
       while (i < n) {
         j = i;
         for (k = i + 1; k < n; k++)
           if (t[k] < t[j])
             j = k;
         w = t[j];
         t[j] = t[i];
         t[i] = w;
         i = i + 1;
       }
     return;
     }
  }
```

1.1 تحليل الخوارزمية

بسبب تكافؤ الخوارزميتين السابقتين، يمكن اعتماد أي منهما من أجل قياس التعقيد الزمني لخوارزمية الفرز بالاختيار الطبيعي. لحساب تعقيد هذه الخوارزمية، سنحسب عدد عمليات المقارنة، وعدد عمليات التبديل.

أولاً: عدد المقارنات

لفرز سلسلة تحوي n عنصراً نحتاج إلى n-1 مقارنة لإيجاد أصغر عنصر، ثم نطبق الطريقة نفسها على سلسلة تحوي n-1 عنصراً. لدينا إذن:

$$Maxc(n) = n-1 + Maxc(n-1) \; ; \; n > 1$$
$$Maxc(1) = 0$$

ولما كان عدد المقارنات لا يتعلق بمحتوى السلسلة فإن:

$$Avc(n) = Maxc(n)$$

إن حل المعادلة التراجعية هو:

$$\text{Maxc(n)} = \text{Maxc(1)} + \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = O(n^2)$$

إذن التعقيد الزمني لخوارزمية الفرز بالاختيار من حيث عدد المقارنات هو:

$$\text{Avc(n)} = \text{Maxc(n)} = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = O(n^2)$$

ثانياً: عدد التبديلات

هنا أيضاً نلاحظ أن عدد التبديلات لا يتعلق بمحتوى السلسلة، إذ نبدل في كل استدعاء عودي للإجرائية عنصرين. إذن :

Av E(n) = Max E(n)
Max E(n) = 1 + Max E(n-1) for i >1
Max E(1) =0
Av E(n) = Max E(n) = n-1 = $O(n)$

2 . الفرز بالتبديل المباشر (الفرز بطريقة الفقاعات Bubble Sort)

بالنظر إلى النسخة التكرارية لخوارزمية الفرز بالاختيار، نلاحظ أنه لإيجاد أصغر عنصر في السلسلة، يمكن مسح السلسلة ابتداء من نهايتيها، وتبديل العناصر المتجاورة إذا كانت لا تراعي الترتيب. في نهاية عملية المسح والتبديل هذه يكون أصغر عنصر موجوداً في بداية السلسلة، ونطبق الطريقة نفسها على الجزء الباقي من السلسلة.

يعود اسم هذه الطريقة إلى صعود أخف (أصغر) العناصر إلى بداية السلسلة، كما تصعد الفقاعات إلى الأعلى في أنبوب اختبار.

مثال

يبين الشكل التالي مثالاً للفرز بطريقة الفقاعات.

Data	130	120	30	62	40	20
Exchanges					20 ↔	40
				20 ↔	62	
			20 ↔	30		
		20 ↔	120			
	20 ↔	130				
Data	20	130	120	30	62	40
Exchanges					40 ↔	62
			30 ↔	120		
		30 ↔	130			

Data	20	30	130	120	40	62
Exchanges			 40 ↔	40 ↔ 130	120	
Data	20	30	40	130	120	62
Exchanges				 62 ↔	62 ↔ 130	120
Data	20	30	40	62	130	120
Exchanges					120 ↔	130
Data	20	30	40	62	120	130

يمكن التعبير عن الفرز بطريقة الفقاعات بالخوارزمية التالية:

```
public static void Buble_Sort (int[] t)
  {
    int i, j, w;
    int n = t.length;


    i = 0;
    while (i < n)
    {
      for (j = n-1; j > i ; j--)
        if (t[j] < t[j - 1])
        {
          w = t[j - 1];
          t[j - 1] = t[j];
          t[j] = w;
        }
      i = i + 1;
    }
    return;
  }
```

2. 1 تحليل الخوارزمية

أولاً: عدد المقارنات

إن عدد المقارنات هو نفسه في خوارزمية الفرز بالاختيار. إذن :

$$\text{Avc(n)} = \text{Maxc(n)} = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = O(n^2)$$

ثانياً: عدد التبديلات

آ- في أسوأ الأحوال لدينا n-1 تبديلاً في الخطوة الأولى و n-2 تبديلاً في الخطوة الثانية وهكذا ...

ونصادف هذه الحالة عندما تكون السلسلة مرتبة ترتيباً تنازلياً. لدينا:

$$\text{MaxE(n)} = \sum_{i=1}^{n-1} (n-i) = \frac{n(n-1)}{2} = O(n^2)$$

ب- في أحسن الأحوال إذا كانت السلسلة مرتبة لا نقوم بأي تبديل :

MinE(n) = 0

ج- العدد الوسطي للتبديلات:

لتكن t سلسلة طولها n نسمي خيال t السلسلة 't التي تحقق:

t'[1] = t[n], t'[2] = t[n-1], ... , t'[n] = t[1]

إذا طبقنا الخوارزمية على كلٍ من t و’t فإن أي زوج من العناصر سيخضع لعملية تبديل أثناء فرز t أو أثناء فرز 't . فالتباديل لدينا بعدد الأزواج المختلفة من العناصر ذات الأدلة المختلفة، أي $\frac{n(n-1)}{2}$ تبديلاً لفرز t و't .

لتكن T مجموعة كل السلاسل التي طولها n والتي تحوي عناصر مختلفة، لنفترض أن كل عناصر T متساوية الاحتمال. فحسب تعريف التعقيد الوسطي، إذا كان C(t) عدد التبديلات اللازمة لفرز السلسلة t و p(t) احتمال السلسلة t فإن:

$$\text{AVE(n)} = \sum_{i \in T} p(t).C(t)$$

يمكن تجزئة المجموعة T إلى جزأين:

Tc : مجموعة السلاسل التي طولها n و تحقق t[1]<t[n]

Td : مجموعة السلاسل التي طولها n و تحققt[1]>t[n]

من الواضح أنه إذا كان t ∈Tc فإن t'∈Td وبالعكس. إذن :

$$\text{AVE(n)} = \sum_{t \in Tc} p(t).C(t) + \sum_{t \in Td} p(t).C(t)$$

وبسبب أن جميع السلاسل متساوية الاحتمال، سنفترض أن p(t) = α

$$\text{AVE(n)} = \alpha\left(\sum_{t \in Tc} C(t) + \sum_{t \in Td} C(t)\right)$$

$$\text{AVE(n)} = \alpha\left(\sum_{t \in Tc}\left(C(t) + C(t')\right)\right)$$

$$\text{AVE(n)} = \alpha\left(\sum_{t \in Tc}\frac{n(n-1)}{2}\right)$$

$$\text{AVE(n)} = \alpha\left(Tc\left(\frac{n(n-1)}{2}\right)\right)$$

و لأنه يوجد تقابل بين Tc و Td فإن $\alpha|Tc| = \frac{1}{2}$

$$\mathrm{AVE(n)} = \frac{n(n-1)}{4} = O(n^2)$$

3. الفرز بالإقحام

نفترض في الخطوة رقم i أن العناصرالـ i – 1 الأولى من السلسلة مرتبة، ونوجد موقع العنصر رقم i بين هذه العناصر.

مثال

يبين الشكل التالي مثالاً للفرز بطريقة الإقحام.

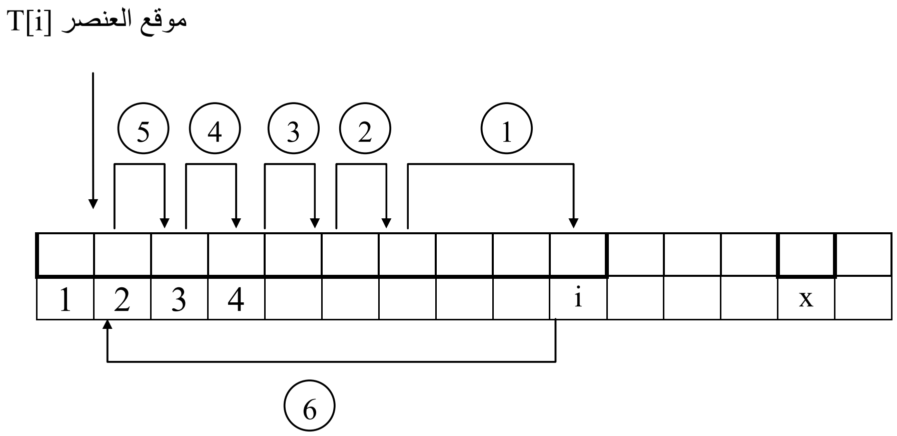
130	120	30	62	40	20
120	130	30	62	40	20
30	120	130	62	40	20
30	62	120	130	40	20
30	40	62	120	130	20
20	30	40	62	120	130

يمكن تحقيق عملية الإقحام بعدة طرق نذكر منها: الفرز بالإقحام التسلسلي والفرز بالإقحام الثنائي.

3. 1 الفرز بالإقحام التسلسلي

نقول إن الإقحام تسلسلي إذا كانت عملية البحث عن موقع العنصر الجديد تجري بالتسلسل. فإذا كانت t قائمة تمثل السلسلة المراد فرزها، فإن عملية إقحام العنصر رقم i تجري بالبحث عن موقع t[i] ضمن القائمة الجزئية t[0, … , i-1] حيث نبدأ من نهايتيها.

تسمح هذه الطريقة بإجراء عمليات الإزاحة الضرورية لعملية الإقحام كما يبين ذلك الشكل التالي.



يمكن التعبير عن طريقة الفرز بالإقحام التسلسلي بالخوارزمية التالية:

```
public static int  range (int[] t,int p,int q,int x)
 {
   int m,r;
     if (p == q)
       r = p;
      else
      {
          m = (p + q)/2;
           if ( x < t[m] )
             r = range(t, p, m, x);
     else
         r = range(t, m, q, x);
      }
     return r;
 }
```

```
 public static int[] bin_insert_sort (int[] t, int i)
 {
   int j, k, x;


   if (i >=1)
   {
     bin_insert_sort(t, i - 1);
     if (t[i - 1] > t[i])
     {
       k = range(t, 0, i - 1, t[i]);
       x = t[i];
       for (j = i - 1; j < k; j--)
         t[j + 1] = t[j];
       t[k] = x;


     }
   }
   return t;
 }
```

تحليل الخوارزمية

أولاً: عدد المقارنات

في أسوأ الأحوال يجب تنفيذ i عملية مقارنة بعد استدعاء Insert_sort (t, i-1) (مقارنة t[0] مع t[k] حيث k = i-1, i-2, ... ; 0). إذن:

$$\text{Maxc}(n) = n + \text{Maxc}(n-1) \quad \text{for } i > 1$$
$$\text{Maxc}(1) = 0$$

ومنه

$$\text{Maxc}(n) = \sum_{i=2}^{n} i = \frac{n(n+1)}{2} - 1 = O(n^2)$$

في أحسن الأحوال لدينا مقارنة واحدة بعد كل استدعاء عودي. أي

$$\text{Minc}(n) = n-1$$

يمكن حساب العدد الوسطي للمقارنات بطريقة مشابهة للطريقة المتبعة في خوارزمية الفرز بطريقة الفقاعات فنجد أن :

$$\text{Avc(n)} = \frac{n(n-1)}{4} = O(n^2)$$

ثانياً: عدد عمليات النسب

في كل مقارنة لدينا عدد عمليات النسب = عدد عمليات المقارنة + 1

إذن:

$$\text{MaxT}(1) = \frac{n(n+1)}{2} = O(n^2)$$
$$\text{MinT}(n) = n$$
$$\text{AvT}(n) = \frac{n(n-1)}{4} + 1 = O(n^2)$$

3. 2 الفرز بالإقحام الثنائي

تتميز هذه الخوارزمية عن سابقتها بأنها تستفيد من كون جزء من السلسلة مرتباً من أجل إيجاد موقع العنصر الجديد.

في المرحلة رقم i تكون السلسلة الجزئيةt[0]، t[2]، ... ، t[i-1] مرتبة تصاعدياً. وللبحث عن موقع العنصر t[i] نقارنه بـ t[i div 2] ونتابع البحث في القسم الأول، أو القسم الثاني من السلسلة الجزئية حسب نتيجة المقارنة.

تصبح خوارزمية الفرز كالتالي:

```
public static int[] bin_insert_sort (int[] t, int i)
 {
   int j, k, x;


   if (i >=1)
   {
     bin_insert_sort(t, i - 1);
     if (t[i - 1] > t[i])
     {
       k = range(t, 0, i - 1, t[i]);
       x = t[i];
       for (j = i - 1; j < k; j--)
         t[j + 1] = t[j];
       t[k] = x;


     }
   }
   return t;
 }

public static int place (int[] t, int i, int j,int k)
 {


   int p, l, w;
   l = i+1;
   k = j;
   while (l <= k) {
     while (t[k] > t[i])
       k = k - 1;
     while (t[l] <= t[i])
       l = l + 1;
     if (l < k)
     {
       w = t[l];
       t[l] = t[k];
       t[k] = w;
       k = k - 1;
       l = l + 1;
     }
   }
   w = t[i];
   t[i] = t[k];
   t[k] = w;
```

```
  return k;

 }
```

تحليل الخوارزمية

إن عدد التبديلات هو نفسه كما في خوارزمية الفرز بالإقحام التسلسلي، ولكن عدد المقارنات يتحسن تحسناً ملحوظاً. في الحقيقة، إن البحث عن موقع عنصر في سلسلة تحوي k عنصراً يتطلب:

Maxrange(k) = $\lceil \log_2 k \rceil$

عملية مقارنة، ومن تتبع آلية تنفيذ الخوارزمية نلاحظ أن:

$$Maxc(n) = 1 + Maxc(n-1) + Maxrange(n-1) \quad \text{for } n > 1$$

ومنه نحصل على العلاقة التراجعية:

$$Maxc(n) = 1 + Maxc(n-1) + \lceil \log2(n-1) \rceil \qquad \text{for } n > 1$$

$$Maxc(1) = 0$$

وبحل هذه العلاقة نحصل على:

$$Maxc(n) = (n-1) + \sum_{k=1}^{n-1} \lceil \log_2 k \rceil = O(n^2)$$

4. الفرز بالتجزئة (الفرز السريع)

خوارزمية الفرز السريع (Quicksort) هي مثال للفرز المعجمي: نقسم سلسلة العناصر المطلوب فرزها إلى سلسلتين جزئيتين، بحيث يكون كل عناصر السلسلة الجزئية الأولى أصغر من كل عناصر السلسلة الجزئية الثانية، نطبق العملية نفسها على السلسلتين الجزئيتين حتى نحصل على سلسلة مؤلفة من عنصر واحد .

لتقسيم السلسلة إلى جزأين يحققان الشرط المطلوب، نختار أحد عناصر السلسلة ونسميه موجهاً (Pivot)، وننشىء سلسلة جزئية كل عناصرها أصغر أو تساوي هذا الموجه، وسلسلة جزئية كل عناصرها أكبر تماماً من الموجه.

إن إنشاء السلسلتين الجزئيتين على السلسلة نفسها (دون استعمال مصفوفة مساعدة) يؤدي إلى وضع العنصر الموجه في المكان النهائي الذي يجب أن يبقى فيه حتى نهاية الفرز.

نحتاج إلى إجرائية

```
place(t, i, j, k)
```

تقوم بتجزئة السلسلة

t[i], …. , t[j]

تبعاً للموجه t[i] وتعطي بالنتيجة الدليل k حيث وضع الموجه.

1 .4 خوارزمية التجزئة

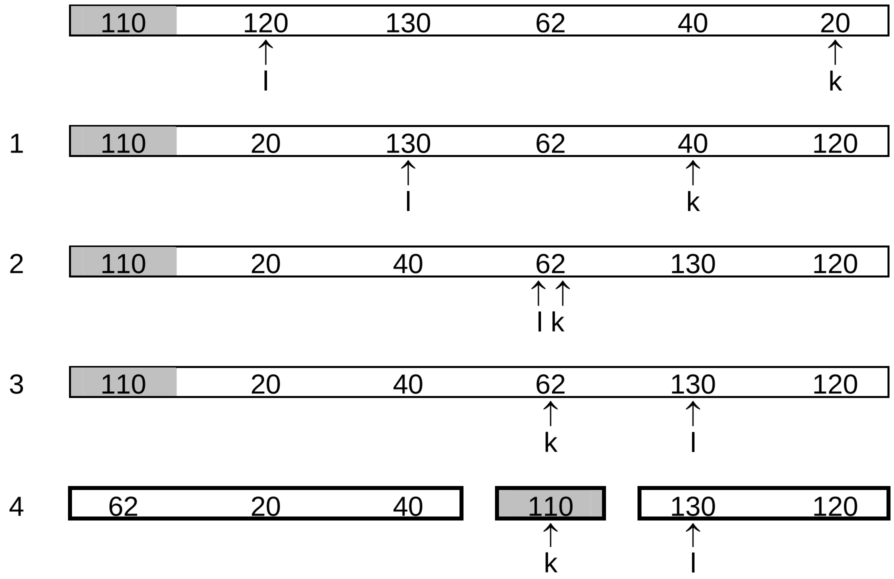
نستعمل دليلين K و L يبدأ كل منهما من أحد أطراف السلسلة ويتجه باتجاه الآخر:

L : يبدأ من بداية السلسلة، ويزداد حتى يصل إلى عنصر أكبر من الموجه.

K : يبدأ من نهاية السلسلة، ويتناقص حتى يصل إلى عنصر أصغر أو يساوي الموجه.

نبدل العنصرين a[L] و a[K] ونتابع العملية السابقة حتى يتقاطع L و K .

مثال



يمكن التعبير عن خوارزمية التجزئة بالإجرائية التالية:

```
public static int place (int[] t, int i, int j,int k)
 {
   int p, l, w;
   l = i+1;
   k = j;
   while (l <= k) {
     while (t[k] > t[i])
       k = k - 1;
     while (t[l] <= t[i])
```

```
       l = l + 1;
     if (l < k)
     {
       w = t[l];
       t[l] = t[k];
       t[k] = w;
       k = k - 1;
       l = l + 1;
     }
   }
   w = t[i];
   t[i] = t[k];
   t[k] = w;
  return k;
 }
```

وتصبح إجرائية الفرز السريع كالتالي:

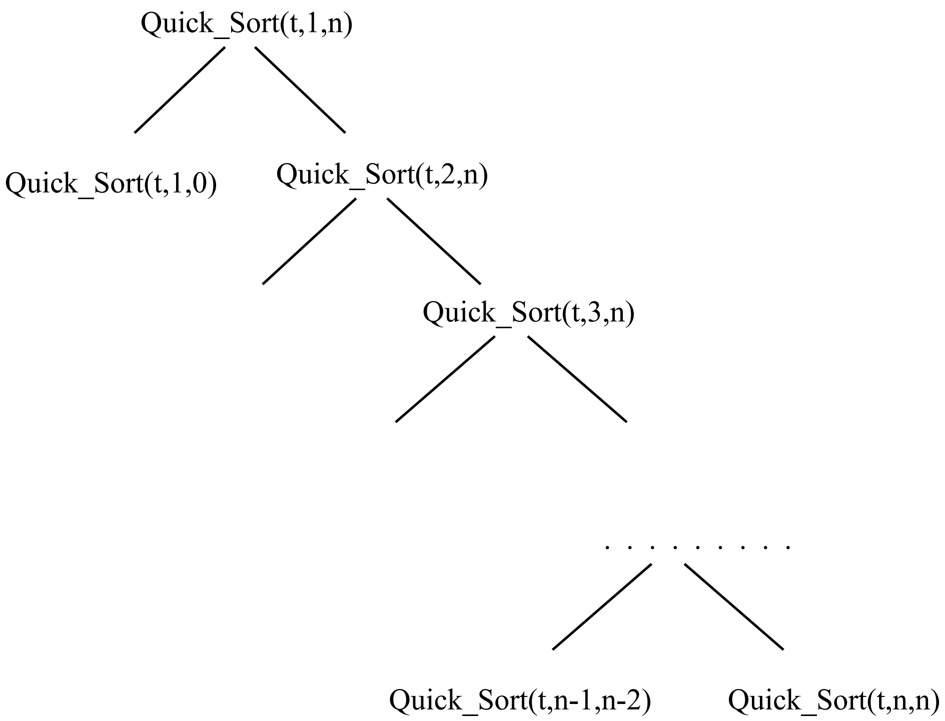
```
public static void Quick_Sort (int[] t, int i, int j)
     {
       int k=0;
   if (i < j) {
    k= place(t, i, j,k);
     Quick_Sort(t, i, k - 1);
     Quick_Sort(t, k + 1, j);


   }
return ;
 }
```

تحليل الخوارزمية

أولاً: عدد المقارنات

لتجزئة سلسلة من n عنصراً نحتاج إلى n+1 مقارنة أو إلى n-1 مقارنة (حسب طريقة تلاقي الدليلين l, k)، ونحتاج إلى n/2 عملية تبديل عناصر. يبين الشكل التالي أسوأ حالة لخوارزمية الفرز السريع.



في هذه الحالة يكون عدد المقارنات :

$$Max_c(n) = (n+1) + (n) + ... + 3 = \frac{(n+2).(n+1)}{2} - 3 = O(n^2)$$

ثانياً: العدد الوسطي للمقارنات

لنفرض أن الموجه يأخذ الموقع رقم p من السلسلة المطلوب فرزها، وأن جميع العناصر مختلفة.

$$n - 1 + \frac{1}{n}.\sum_{p=1}^{n}\left(Av_c(p-1) + Av_c(n-p)\right) \le Av_c(n)$$

$$Av_c(n) \le n + 1 + \frac{1}{n}.\sum_{p=1}^{n}\left(Av_c(p-1) + Av_c(n-p)\right)$$

$$Av_c(0) = Av_c(1) = 0$$

$$A(n) \le Av_c(n) \le B(n)$$

حيث

$$A(n) = n - 1 + \frac{1}{n}.\sum_{p=1}^{n}\left(A(p-1) + A(n-p)\right) \; B(n) = n + 1 + \frac{1}{n}.\sum_{p=1}^{n}\left(B(p-1) + B(n-p)\right)$$

$$A(0) = B(0) = 0$$

$$A(n) = n - 1 + \frac{2}{n} . \sum_{p=1}^{n-1} (A(p))$$

$$B(n) = n + 1 + \frac{1}{n} . \sum_{p=1}^{n-1} (B(p))$$

حل هذه المعادلات التراجعية هو:

$$A(n) = 2(n+1)H_n - 4n$$

$$B(n) = 2(n+1)\left(H_n - \frac{4}{3}\right) = 2(n+1)H_n - \frac{8}{3}n - \frac{8}{3}$$

حيث

$$H_n = \sum_{i=1}^{n} \frac{1}{i} \approx Logn$$

إذن:

$$Av_c(n) \approx 2n.Logn \approx 1{,}38n.\log_2 n$$

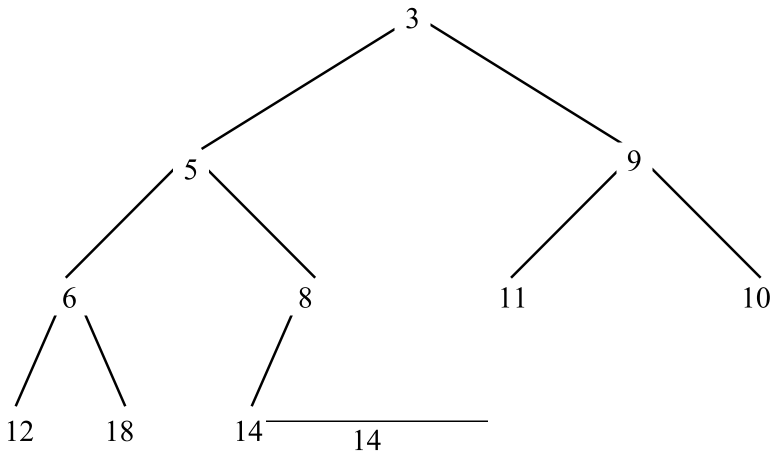
5. الفرز باستعمال الأشجار المرتبة جزئياً

الشجرة المرتبة جزئياً هي شجرة تحقق الشرط التالي:

إذا كانت o عقدة و $f1, f2, \ldots, fn$ أولاد العقدة o فإن $o \le fi$ من أجل $i = 1, 2, \ldots , n$

أما الكومة فهي شجرة ثنائية كاملة مرتبة جزئياً .

مثال



ولما كانت الكومة شجرة ثنائية كاملة، أمكن تمثيلها بواسطة مصفوفة t بحيث:

t[1]   يحوي الجذر

t[i div 2] يحوي العقدة الأم للعقدة i وذلك في حال i > 1

ونستعمل عداداً p يحوي عدد عقد الشجرة.

بهذا يمكن عد الكومة متتالية t[1], t[2], ..., t[p] تحقق الشرط:

في حال i ≤ p div 2

t[i] ≤ t[2*i]
t[i] ≤ t[2*i +1]

1 .5 استعمال الكومة في الفرز

بافتراض أن العناصر المراد فرزها موجودة في كومة، فإن أصغر عنصر سيكون في القمة، نحذف هذا العنصر ونعيد بناء الكومة. تجري عملية الحذف وإعادة بناء الكومة كما يلي:

- نبدل t[p] بـ t[1]
- نُنقص عدد عناصر الكومة بـ 1
- نناقش: هل يحقق العنصر الجديد الذي يشغل قمة الكومة شروط الكومة ؟

إذا اختل الشرط فإننا نبدل هذا العنصر بأصغر ولديه ونتابعه في موقعه الجديد.

يمكن التعبير عن عملية إعادة بناء الكومة بالإجرائية التالية:

```
public static void  sift (int l,int  r)
 {
 int i, j;
 int [] t=new int [6];
  int x;
   i = l;
   j = 2 * i;
   while (j <= r)
   {
     if (j < r)
       if (t[j] > t[j + 1])
         j = j + 1;
     if (t[i] <= t[j])
       j = r + 1;
     else {
       x = t[i];
       t[i] = t[j];
       t[j] = x;
       i = j;
     }
   }
 return;
 }
```

5. 2 بناء الكومة

يمكن استعمال الإجرائية السابقة نفسها لبناء الكومة، وبافتراض أن العدد الكلي للعناصر هو n ، يمكن بناء الكومة بتنفيذ التعليمات:

```
  int l = (n / 2) + 1;
  int  r = n;
   while (l > 1) {
     l = l - 1;
     sift(l, n);
   }
```

بذلك تصبح إجرائية الفرز باستعمال كومة كالتالي:

```
public static void  Heap_sort (int[] t)
 {
   int n = t.length;
   int l = (n / 2) + 1;
   int  r = n;
   while (l > 1) {
     l = l - 1;
     sift(l, n);
   }
   while (r > 1)
    {
     int   x = t[1];
       t[1] = t[r];
       t[r] = x;
       r = r - 1;
       sift(1, r);
     }
   return;
 }
```

تحليل خوارزمية الفرز باستعمال كومة

نترك للطالب إيجاد التعقيد الزمني لهذه الخوارزمية في الفرز، من حيث عدد المقارنات وعدد التباديل.

إن الدراسة المتأنية لمراحل تنفيذ الخوارزمية تبين أن التعقيد الزمني لهذه الخوارزمية هو من مرتبة n log(n) في أسوأ الأحوال.

6. تمرينات

1- نقول عن طريقة ما في الفرز إنها مستقرة إذا كانت تحافظ على الترتيب النسبي للعناصر المتساوية المفتاح.

مثال

إذا كانت لدينا قائمة من العناصر T تحوي تسجيلات تتألف كل منها من اسم الشخص وعمره، وكان لدينا قبل الفرز T[i].age = t[j].age حيث i<j وأخذ العنصران i و j الموقعين k و l على الترتيب بعد الفرز، فيجب أن يكون k < l.

آ- برهن أن طريقة الفرز بالفقاعات مستقرة.

ب- اعط مثالاً يبين أن طريقة الفرز السريع غير مستقرة.

2- الغرض من هذا التمرين دراسة حالات خاصة لعمليات الفرز.

آ- لدينا مجموعة من الكرات عددها **n** ، لكل منها رقم ولون من ثلاثة ألوان هي الأحمر **(R)** والأسود **(B)** والأبيض **(W)** ، نريد فرز هذه الكرات بحيث نحصل على الكرات الحمر أولاً والبيض ثانياً والسود آخراً، يشترط في خوارزمية الفرز أن تقوم بفحص لون الكرة مرة واحدة فقط. اكتب هذه الخوارزمية.

ب- كيف يمكن تعميم طريقة الفرز السابقة حين يكون عدد الألوان أكبر من 3 ؟

ج- بين كيف يمكن الاستفادة من طريقة الفرز السابقة في فرز مجموعة من أوراق اللعب، وفق اللون (مفتاح أولي) والقيمة (مفتاح ثانوي)، مع العلم أن ترتيب الألوان هو:

♠ > ♥ > ♦ > ♣ وأن ترتيب القيم هو :

A > K > Q > J > 10 > 9 > 8 > 7 > 6 > 5 > 4 > 3 > 2

3- اكتب نسخة محسنة لخوارزمية الفرز بطريقة الفقاعات تأخذ بعين الاعتبار الملاحظات التالية :

آ- يمكن لخوارزمية الفرز بطريقة الفقاعات أن تمسح السلسلة دون القيام بأي تبديل.

ب- لتجنب المقارنات غير الضرورية حين يكون القسم الأول من السلسلة مرتبا، يمكن الاحتفاظ بدليل آخر عنصر حصل عنده تبديل (السلسلة الجزئية المؤلفة من العناصر التي تسبق هذا الدليل تكون مرتبة).

ج- هناك عدم تناظر في تنفيذ الخوارزمية، من حيث عدد المراحل اللازمة للفرز، نلاحظ ذلـك بتطبيق الخوارزمية على السلسلتين :

2, 3, 4, 5, 6, 7, 8, 1
8, 1, 2, 3, 4, 5, 6, 7

في الحالتين هناك عنصر واحد يخالف الترتيب، ولكن يلزم طور واحد لفرز السلسلة الأولى وسبعة أطوار لفرز السلسلة الثانية.

لحل هذه المشكلة نجعل المرور على عناصر السلسلة يجري مرة تنازلياً، وأخرى تصاعدياً حتى نهاية الفرز.

4- ما هو عدد المقارنات والتبديلات التي تجريها خوارزمية الفرز السريع عند تطبيقها على سلسلة جميع عناصرها متساوية.