# البحث الثنائي

# Binary Search

## تعريف :

عندما يُطلب من أحدنا البحث عن شيء فإنه يبدأ من الصفر إلى النهاية كي يجده و يعتقد أنه لن يستطيع توفير أي وقت في البحث و ذلك لأنه في حال غض النظر عن أي مكان فإنه من الممكن أن يكون هنالك الشيء الذي تبحث عنه .
بكلام برمجي أخر ، عندما نبحث في اللائحة فإننا نبحث و نحن على يقين بأنها عشوائية ، لذا فلا يوجد حل إلا بالبحث المتتالي و الذي يجب أن يكون من البداية إلى النهاية أو حتى إيجاد المفتاح ( العنصر الذي نبحث عنه )

```
For(i=0 ; i<n; i++)
{
        if(key==a[n])
          return n;
}
```

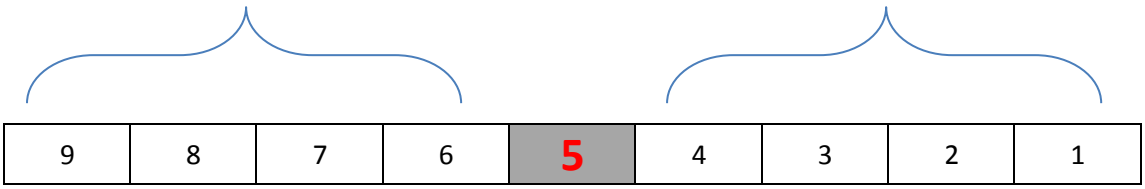
و عندما يصل العداد إلى n فهذا يعني أنه لم يعثر على المفتاح

و لكن أحياناً تكون اللائحة ضخمة جداً و يصبح من الصعب البحث في كل حجرة ، لذا فمن الأولى أن نطبق البحث الثنائي :

1- أولاً ... يتم ترتيب العناصر فيها ( تصاعدياً أو تنازلياً )
2- تعيين العنصر الذي في المتنصف mid و يسمى الجذر root و ذلك عن طريق المتوسط الحسابي لأكبر و أصغر رقمين
3- تقسيم اللائحة إلى قسمين ، قسم أكبر من العنصر mid " يسار " و قسم أصغر من العنصر mid " يمين "
4- تعيين جذر لكل غصن ( قسم أكبر و قسم أصغر ) و تكرار ما تم فعله في الخطوة2 حتى نصل إلى أخر و أول عنصرين

نحسب المتوسط الحسابي لأكبر و أصغر رقم في كل قائمة ، High=9 , Low=1

$$\text{Root} = \frac{1+9}{2} = 5$$





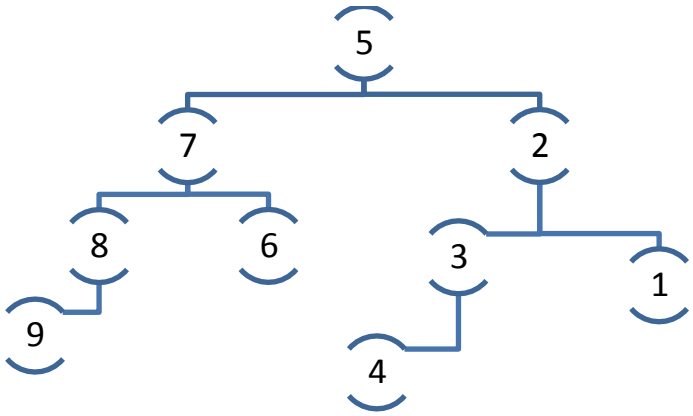

$$\text{Root} = \frac{1+4}{2} = 2.5$$

نقوم بتقريب الرقم و يجوز إلى 2 و يجوز إلى 3 ، يفضل 2 لأنها أسهل برمجياً

$$\text{Root} = \frac{6+9}{2} = 7.5$$

نقوم بتقريب الرقم و يجوز إلى 7 و يجوز إلى 8 ، لكن سنختار 7 لنفس السبب السابق

في رسم الشجرة الثنائية ، نضع الجذر في المنتصف و الأعداد الأكبر في الغصن اليساري و الأعداد الأصغر في الغصن اليميني



# الكود بطريقة تكرارية

```
int search(int a[],int key,int hi,int lo)   //hi=array.length  &  lo=0
{


        int m;
        while(hi>=lo)
        {
         m=(hi+lo)/2;
         if(key<a[m])
         {
               hi=m-1;

         }
        else if(key>a[m])
         {
               lo=m+1;

         }
        else
         return m;
        }
        return -1;

}
```

# الكود بطريقة تعاودية

```
int search(int a[],int key,int hi,int lo)
{


        int m;
        if(hi<lo)
               return -1;
        else
        {
         m=(hi+lo)/2;
        if(key<a[m])
         {
               hi=m-1;
               search(a,key,hi,lo);
         }
        else if(key>a[m])
         {
               lo=m+1;
               search(a,key,hi,lo);
         }
        else
         return m;
        }

}
```

# التحليل الزمني:

## في أفضل الأحوال :

اذا كان العنصر المراد البحث عنه في منتصف المصفوفة

$$O(n)=1$$

## في أسوء الأحوال :

اذا كان العنصر المراد البحث عنه هو أكبر قيمة \ أو أصغر قيمة في المصفوفة

سنناقش حالة العدد هو العدد الأصغري في المصفوفة :

$$hi = n$$

$$lo=0$$

$$m = \frac{hi}{2}$$

$$hi = m - 1$$

نجرب بطريقة تكرارية

$$hi = m - 1 \qquad m = \frac{m-1}{2}$$

$$hi = \frac{m-3}{2} \qquad m = \frac{m-3}{4}$$

$$hi = \frac{m-7}{4} \qquad m = \frac{m-7}{8}$$

$$hi = \frac{m-15}{8} \qquad m = \frac{m-15}{16}$$

نستنتج الشكل العام لـ n :

$$\mathbf{T(n) = \frac{m-(2^{k+1})-1}{2^k}}$$

$$\mathbf{T(n) = \frac{\frac{n}{2} - (2^{k+1}) - 1}{2^k}}$$

بما أن الحلقة التكرارية لا تنتهي في أسوء الأحوال إلا اذا كان lo>=hi و في حالتنا يجب أن يكون العدد hi صفر أو سالب كي تنتهي الحلقة . سنستخدم المساواة لأنه التابع تناقصي و اذا وصل للصفر فسيستمر و يصبح سالب .

$$\frac{n}{2} - (2^{k+1}) - 1 = 0$$

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

$$n = (2^{k+2}) + 2$$

$$n - 2 = 2^{k+2}$$

$$\frac{n-2}{4} = 2^{k}$$

$$k = \log_2(\frac{n-2}{4})$$

اذاً التعقيد لخوارزمية البحث الثنائي لوجاريتمي :

$$\mathbf{O(n) = log(n)}$$

أما بالنسبة للبحث الخطي فالتعقيد خطي لأنه حلقة تكرارية وحيدة طولها n

$$\mathbf{O(n) = n}$$