فعالية الخوارزميات Algorithm Efficiency
وخوارزميات البحث Search Algorithms

إعداد المهندس: يوسف دعبول

فعالية الخوارزمية Algorithm Efficiency

- يتم تحديد درجة تعقيد الخوارزمية بالاعتماد على عدة عوامل:
  1. حجم الذاكرة اللازم لتخزين المعطيات المراد معالجتها
  2. الزمن اللازم لإدخال المعطيات إلى الذاكرة
  3. الزمن اللازم لتنفيذ الخوارزمية

وهنا نهمل دراسة عامل حجم الذاكرة لأهمية عامل الزمن

عامل الزمن

- يعتمد عامل الزمن على عدة عوامل منها:
  1. حجم الدخل: حيث يتطلب زمن لإدخال العناصر إلى الذاكرة والتعبير عنها
  2. عدد العناصر: ترتيب عناصر لائحة ما يتعلق بعدد العناصر

أي أن زمن التنفيذ **تابع** لعدد العناصر n

أي زمن التنفيذ هو T(n)

لا يمكن تحديد زمن التنفيذ بوحدات الزمن

- لأن زمن التنفيذ يعتمد على:
  1. نوع التعليمات
  2. نوع الجهاز المستخدم
  3. السرعة التي تنفذ بها الآلة

وبالتالي زمن التنفيذ T(n) يعبر عن عدد مرات تنفيذ الخوارزمية

حساب الزمن لخوارزمية حساب القيمة المتوسطة

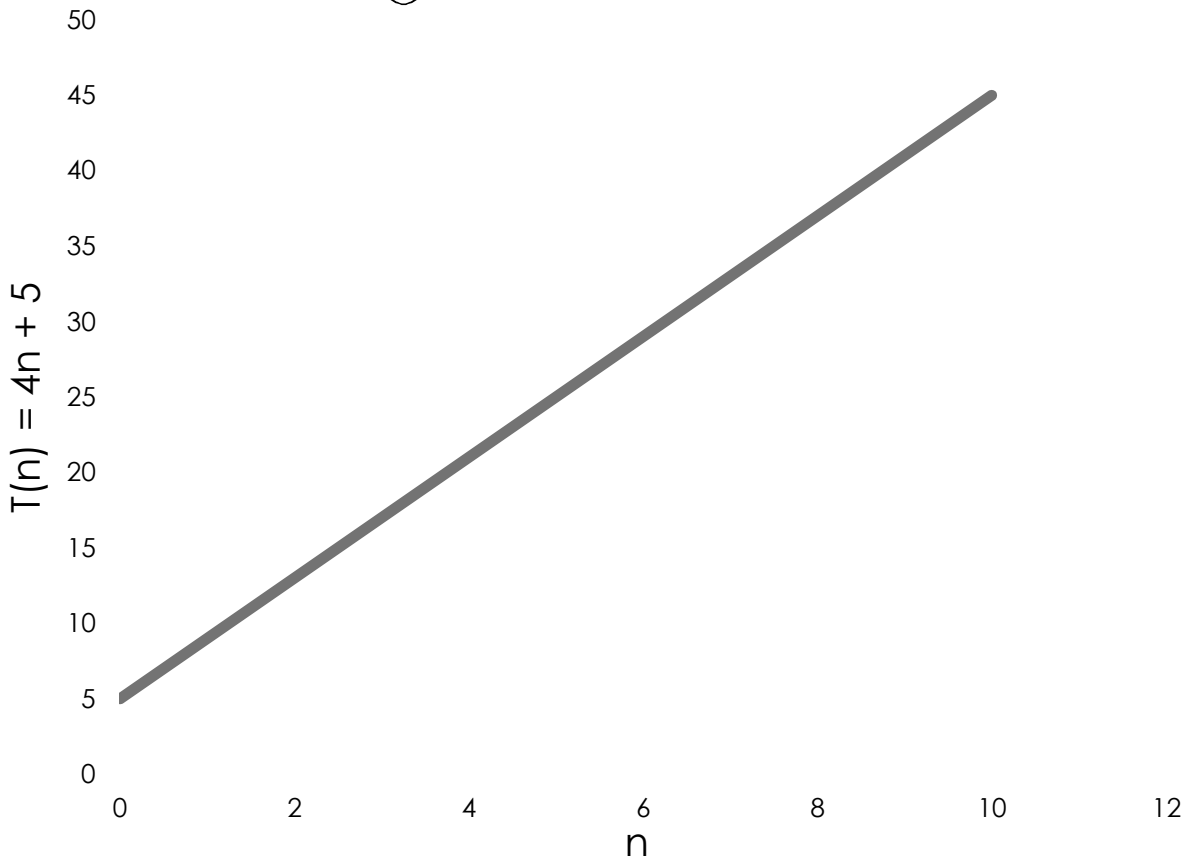
1. Read n
2. Sum = 0
3. i = 1
4. While i <= n do
   A. Read Number
   B. Sum = Sum + Number
   C. i = i + 1
5. Calculate Mean = Sum / n

حساب الزمن لخوارزمية حساب القيمة المتوسطة

1. Read n	1
2. Sum = 0	1
3. i = 1	1
4. While i <= n do	n + 1
A. Read Number	n
B. Sum = Sum + Number	n
C. i = i + 1	n
5. Calculate Mean = Sum / n	1

$T(n) = 4n + 5$

حساب الزمن لخوارزمية حساب القيمة المتوسطة



$T(n) = 4n + 5$

علاقة زمن التنفيذ هي تابع خطي لعدد العناصر n

نقول زمن تنفيذ الخوارزمية T(n) تابع من مرتبة n

ونعبر عنه $T(n) = O(n)$

خوارزميات البحث

- بفرض لدينا قائمة بطلاب صف ما
- وأردنا البحث عن طالب برقمه الجامعي
- تهمنا أن تكون الخوارزمية ذات **فعالية عالية** خاصة إذا كانت قائمة الطلاب كبيرة

- من الخوارزميات المتبعة:
  - البحث الخطي (التتابعي) Linear (Sequential) Search
  - البحث الثنائي Binary Search

خوارزمية البحث الخطي (التتابعي)

- نقوم بالمرور على جميع عناصر القائمة حتى يتم إيجاد العنصر أو الوصول إلى نهاية القائمة دون إيجاده

- ندرس الحالة الأسوأ وهي عندما لا يتم إيجاد العنصر وبالتالي يتم المرور على كافة عناصر القائمة

خوارزمية البحث الخطي (التتابعي)

المتغيرات: Found ← النتيجة Loc ← موقع العنصر الجاري اختباره

Item ← العنصر المراد البحث عنه

1. Found = False
2. Loc = 1
3. While Loc <= n And Found == False Do
   A. If Item = A[Loc]
      B. Found = True
   Else
      C. Loc = Loc + 1

خوارزمية البحث الخطي (التتابعي)

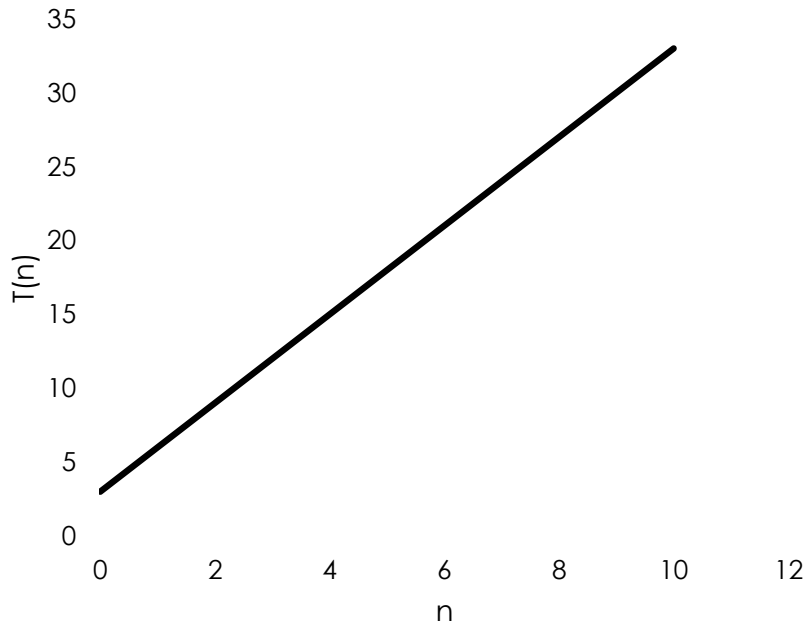
المتغيرات: Found ← النتيجة Loc ← موقع العنصر الجاري اختباره

Item ← العنصر المراد البحث عنه

1. Found = False	1	
2. Loc = 1	1	
3. While Loc <= n And Found == False Do	n + 1	T(n) = 3n + 3
A. If Item = A[Loc]	n	
B. Found = True	0	
Else		
C. Loc = Loc + 1	n	

خوارزمية البحث الخطي (التتابعي)

- $T(n) = 3n + 3$ هو تابع خطي
- ومن هنا جاءت التسمية «البحث الخطي»



خوارزمية البحث الخطي (التتابعي)

```
const int n = 10;
int A[n] = {7, 3, 1, 0, 21, 5, 17, 99, 2, -5};
for (int i=0; i<n; i++) {
        cout<<setw(4)<<A[i];
}
cout<<endl;
for (int i=0; i<n; i++) {
        cout<<setw(4)<<i + 1;
}
cout<<endl;
int found;
int loc;
int item;
cout<<"Input Item to search for: ";
cin>>item;
found = 0;
loc = 0;
while ((loc < n) && (!found)) {
        if (item == A[loc])
                found = 1;
        else
                loc++;
}
if (found)
        cout<<item<<" was found at location #"<<loc + 1;
else
        cout<<item<<" was not found!";
```

خوارزمية البحث الثنائي

- أكثر فعالية من خوارزمية البحث الخطي
- تشترط أن تكون العناصر **مرتبة** تصاعديا أو تنازليا
- مثال: البحث عن الرقم 9

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
3	7	9	10	11	16	20	23	24	25	30	37	40	43	44	45	47	48	50	51	52

3	7	9	10	11	16	20	23	24	25

3	7	9	10

9	10

خوارزمية البحث الثنائي

	1	2	3	4	5	6	7	8	9	10
A	3	7	9	10	11	16	20	23	24	25

1. Found = False
2. First = 1
3. Last = n
4. While First <= Last And Found == False Do
   A. Calc Mid = (First + Last)/2
   B. if key > A[Mid]
      B1. First = Mid + 1
          if(A[First]==key)
            found=true
   C. ElseIf key < A[mid]
      C1. Last = Mid – 1
          if(A[Last]==key)
          found=true
   D. Else Found = True

خوارزمية البحث الثنائي

- الحالة السيئة هي عندما يكون العنصر المراد البحث عنه أكبر أو أصغر من كافة عناصر اللائحة
- مثلا: البحث عن الرقم 1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
3	7	9	10	11	16	20	23	24	25	30	37	40	43	44	45	47	48	50	51	52
3	7	9	10	11	16	20	23	24	25											
3	7	9	10																	
3																				

خوارزمية البحث الثنائي

- الحالة السيئة هي عندما يكون العنصر المراد البحث عنه أكبر أو أصغر من كافة عناصر اللائحة
- مثلا: البحث عن الرقم 1
- K هو عدد مرات التقسيم اللازمة للحصول على عنصر واحد

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
3	7	9	10	11	16	20	23	24	25	30	37	40	43	44	45	47	48	50	51	52
3	7	9	10	11	16	20	23	24	25											
3	7	9	10																	
3																				

- بعد K تقسيم يكون حجم اللائحة $n/2^k$
- الحالة النهائية: $n/2^k < 2$ ← $k+1 > \log_2(n)$ نقربها $k \approx \log_2(n)$

خوارزمية البحث الثنائي

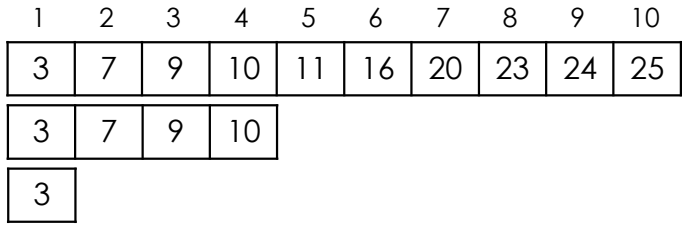
1	2	3	4	5	6	7	8	9	10
3	7	9	10	11	16	20	23	24	25

3	7	9	10

3

1	1. Found = False
1	2. First = 1
1	3. Last = n
1 1 1 1	4. While First <= Last And Found == False Do
1 1 1	A. Calc Mid = (First + Last)/2
1 1 1	B. if key > A[Mid]
	B1. First = Mid + 1
1 1 1	C. Elself key < A[mid]
1 1 1	C1. Last = Mid – 1
	D. Else Found = True

خوارزمية البحث الثنائي



1	1. Found = False
1	2. First = 1
1	3. Last = n
$\log_2(n) + 2$	4. While First <= Last And Found == False Do
$\log_2(n) + 1$	A. Calc Mid = (First + Last)/2
$\log_2(n) + 1$	B. if key > A[Mid]
	B1. First = Mid + 1
$\log_2(n) + 1$	C. Elself key < A[mid]
$\log_2(n) + 1$	C1. Last = Mid – 1
	D. Else Found = True

$9 + 5\log_2(n)$

خوارزمية البحث الثنائي

- وظيفة: اكتب خوارزمية البحث الثنائي بلغة C++

المقارنة بين الخوارزميتين