E-LEARNING REVISION

CE2001/CZ2001 – ALGORITHMS

---

1. Suppose the following algorithm is used to evaluate the following polynomial

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

```
p = a[0];
xpower = 1;
for (i=1; i<=n; i++) {
        xpower = x * xpower;
        p = p + a[i] * xpower;
}
```

(a) How many multiplications are done in the best case, average case and worst case?

(b) Show how you may improve on the algorithm. What will be the number of multiplications in the best case, average case and worst case? Can you improve the algorithm such that the number of multiplications is n in the worst case?

2. Arrange the functions below from lowest asymptotic order to highest asymptotic order. If any two (or more) are of the same asymptotic order, indicate them.

$5n$	2^n	$n\,lg(n)$	$n^{3/2}$	$ln(n)$
$100n^2$	$e\,lg(n)$	$n - n^3 + 7n^5$	$n^2 + lg(n)$	e^n

3. State TRUE or FALSE for each of the following statements. Note that ϵ means "is in" and is equivalent to "=" as used in lecture notes.

(a) If $f(n) \epsilon O(g(n))$, we say that $f(n)$ is the *asymptotic upper bound* for $g(n)$.

(b) $2n^2(n+1) \epsilon \Omega(n^3)$

(c) $log_{10}(n) \epsilon O(log_2(n))$

(d) If $f(n) \epsilon O(g(n))$ and $g(n) \epsilon \Omega(f(n))$, then $f(n) \epsilon \Theta(g(n))$.

(e) $(3.1)^n \epsilon \Theta(3^n)$

4. Show the content of the hash table (with a size of 11) after storing a list of keys {1101, 2973, 2588, 1222, 1343} using each of the following hashing methods.

   (a) Closed address hashing:

   $f(k) = k \bmod 11$

   (b) Open address hashing with double hashing:

   $f(k) = k \bmod 11$, $d = \text{hashIncr}(k) = k \bmod 6 + 1$;
   $\text{rehash}(j, d) = (j+d) \bmod 11$

   where k is the key value, j is the index of the hashed cell, and d is the hash increment.

   In each of the above cases, what is the total number of key comparisons encountered?

5. Note that in Insertion sort, to find the correct position for the ith element, we use a Sequential search (backward) through the sorted subarray $A[1 \,..\, i-1]$. Show how Binary search can be used to reduce the worst-case number of key comparisons of Insertion sort from $\Theta(n^2)$ to $O(n \log_2 n)$.

6. Let $E[1 \,..\, n]$ be an array of n distinct letters which can be ranked in alphabetical order (i.e. $a < b < \ldots < z$). Let i and j be indices of array E. If $1 \le i < j \le n$ and $E[i] > E[j]$, then the pair of indices (i, j) is called an *inversion* of array E.

   (a) Suppose the array E is (c, f, a, h, e, b, d, g), list all inversions in this array.

   (b) Among all arrays with elements from the set $\{1, 2, \ldots, n\}$, which array has the most inversions? How many inversions does it have?

   (c) If Merge sort is applied to the array E given in (a) above, what is the number of comparisons between array elements? Justify your answer.

1. Suppose the following algorithm is used to evaluate the following polynomial

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

```
p = a[0];
xpower = 1;
for (i=1; i<=n; i++) {
        xpower = x * xpower;
        p = p + a[i] * xpower;
}
```

(a) How many multiplications are done in the best case, average case and worst case?

(b) Show how you may improve on the algorithm. What will be the number of multiplications in the best case, average case and worst case? Can you improve the algorithm such that the number of multiplications is n in the worst case?

How algo works?

When i=1 (2 times)

xpower = x

$p = a[0] + a[1]x$

No. of multiplications.

Best case = 2n

Average case = 2n

Worst case = 2n

When i=2 (4 times)

xpower = x^2

$P = (a[0] + a[1]x) + (a[2]x^2)$

When i=2 (8 times)

xpower = x^3

$P = (a[0] + a[1]x) + (a[2]x^2) + (a[3]x^3)$

⋮

(b) Worst case: n

Possible solution: Factorising the polynomial (method 1)

```
p = a[n];
for (i=n; i>=1; i--)
    p = p*x + a[i-1]
```

Best case: n
Average case: n
Worst case: n

using n=3 as e.g.

$p = a[3]$

$\underline{i=3}$

$p = a[3]x + a[2]$

$\underline{i=2}$

$p = (a[3]x + a[2])(x) + a[1]$
$= a[3]x^2 + a[2]x + a[1]$

$\underline{i=1}$

$p = (a[3]x^2 + a[2]x + a[1])(x) + a[0]$
$= a[3]x^3 + a[2]x^2 + a[1]x + a[0]$

<u>method 2:</u>

```
p = a[0]
xpower = 1
for( i=1; i<=n; i++)
    {  xpower = x * xpower;
       if (a[i] <> 0)
           p = p+a[i] * xpower;
    }
```

Best case: n
Average case: 1.5n (assuming 50% chance for a[i] ==0)
Worst case: 2n.

2. Arrange the functions below from lowest asymptotic order to highest asymptotic order. If any two (or more) are of the same asymptotic order, indicate them.

$5n$ 3	2^n 6	$n\ lg(n)$ 4	$n^{3/2}$ 8	$ln(n)$ 1
$100n^2$ 10	$e\ lg(n)$ 2	$n - n^3 + 7n^5$ 11	$n^2 + lg(n)$ 9	e^n 5

$e \doteq 2.7\ldots$

1 $\ln(n)$ SAME
2 $e\lg(n)$ ✓
3 $5n$ ✓

4 $n\lg(n)$ ✓
5 e^n switch. SAME
6 2^n X

7 $n^{3/2}$
8 $n^2 + \lg(n)$ SAME
9 $100n^2$

10 $n - n^3 + 7n^5$

$e\lg n$, $\ln(n)$ – same
$5n$
$n\lg(n)$
$n^{3/2}$
$100n^2$, $n^2 + \lg(n)$ - same
$n - n^3 + 7n^5$
2^n
e^n

3. State TRUE or FALSE for each of the following statements. Note that ϵ means "is in" and is equivalent to "=" as used in lecture notes.

(a) If $f(n) \in O(g(n))$, we say that $f(n)$ is the *asymptotic upper bound* for $g(n)$.

(b) $2n^2(n+1) \in \Omega(n^3)$

(c) $log_{10}(n) \in O(log_2(n))$

(d) If $f(n) \in O(g(n))$ and $g(n) \in \Omega(f(n))$, then $f(n) \in \Theta(g(n))$.

(e) $(3.1)^n \in \Theta(3^n)$

3(a) TRUE ✗ FALSE. It should be $g(n)$ is asymptotic upper bound of $f(n)$

3(b) $2n^2(n+1) \in \Omega(n^3)$

def of Big Ω

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$$

let $f(n) = 2n^2(n+1)$

let $g(n) = n^3$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2n^2(n+1)}{n^3}$$

$$= \lim_{n\to\infty} \frac{2(n+1)}{n}$$

$$= \lim_{n\to\infty} \frac{\cancel{\frac{2n}{n}} + \frac{1}{n}}{\cancel{n/n}}$$

$$= \lim_{n\to\infty} 2 + \frac{1}{n}$$

$$= 2$$

Since $\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$ $\therefore$ $2n^2(n+1) \in \Omega(n^3)$ is TRUE ✓

3c) $log_{10}(n) \in O(log_2(n))$

def of O notation

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\log_{10} n}{\log_2 n}$$

$= \lim \Big(\cancel{\log_2 n} \times \frac{1}{\log ...}$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

$f(n) = \log_{10}(n)$

$g(n) = \log_2(n)$

$$\lim_{n\to\infty} \left(\frac{\quad}{\log_2 10} \times \frac{\quad}{\log_2 n} \right)$$

$$= \frac{1}{\log_2 10}$$

Ans: TRUE

3(d) Big O notation

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

~~True~~

FALSE

Ω Notation

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$$

Θ Notation

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} < C$$

(e) $f(n) = (3.1)^n$

$g(n) = 3^n$

FALSE

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{(3.1)^n}{3^n}$$

$$= \lim_{n\to\infty} \left(\frac{3.1}{3} \right)^n$$

$$= \infty$$

4. Show the content of the hash table (with a size of 11) after storing a list of keys {1101, 2973, 2588, 1222, 1343} using each of the following hashing methods.

(a) Closed address hashing:

$f(k) = k \bmod 11$

(b) Open address hashing with double hashing:

$f(k) = k \bmod 11$, $d = \text{hashIncr}(k) = k \bmod 6 + 1$;
$\text{rehash}(j, d) = (j+d) \bmod 11$

where k is the key value, j is the index of the hashed cell, and d is the hash increment.

In each of the above cases, what is the total number of key comparisons encountered?

(a)

	Hash table	Keys
0		
1	1101	→ 1222 → 1343
2		
3	2973	→ 2588
4		
5		
6		
7		
8		
9		
10		

Total number of key comparisons $= 1 + 1 + 2 = 4$

How to count?

Key space

$f(K) = K \bmod 11$

$f(1101) = 1101 \bmod 11$
$= 1$

$f(2588) = 2588 \bmod 11$
$= 3$

$f(2973) = 2973 \bmod 11$
$= 3$

$f(1222) = 1222 \bmod 11$
$= 1$

d

$f(1343) = 1343 \bmod 11$
$= 1$

Hash table

No. collisions

for the value.

(b) $f(k) = k \bmod 11$

$d = \text{hashIncr}(k)$
$= k \bmod 6 + 1$

$\text{rehash}(j, d) = (j + d) \bmod 11$

0	1222	2
1	1101	0
2		
3	2973	1
4		
5	~~1222~~	~~2~~
6	2588	1
7	1343	1
8		
9		
10		

$f(1101) = 1$

$f(2973) = 3$

$f(2588) = 3 \Rightarrow$ use increment $d = 2588 \bmod 6 + 1 = 2 + 1 = 3 \Rightarrow$ use Rehash func $\text{Rehash} = (j+d) \bmod 11 = (3+3) \bmod 11 = 6 \bmod 11 = 6$

$f(1222) = 1 \Rightarrow$ use increment $d = 1222 \bmod 6 + 1 = 4 + 1 = 5$

$f(1343) = 1 \Rightarrow$ use increment $d = 1343 \bmod 6 + 1 = 5 + 1 = 6 \Rightarrow$ use Rehash function $\text{Rehash} = (6+6) \bmod 11 = 1$

$f(1222) = \cancel{1222 / 11}$
$= 1222 \bmod 11$
$= 1$
~~Rehash=~~
$d = 5$
$\text{Rehash} = (6, 5)$
$= 6 + 5 \bmod 11$
$= 0.$

5. Note that in Insertion sort, to find the correct position for the ith element, we use a Sequential search (backward) through the sorted subarray $A[1 \,..\, i-1]$. Show how Binary search can be used to reduce the worst-case number of key comparisons of Insertion sort from $\Theta(n^2)$ to $O(n \log_2 n)$.

insertion sort

```
void InsertionSort(AList slot[], int n)
{  // inputslot is an array of n records
   // assume n>1
   for (int i=1, i<n; i++)
```

In insertion sort, to find the correct position for the ith element, use Binary search

• subarray $A[1 \ldots i-1]$ is already

```
for(int j=i; j>0; j--)
    if( slot[j].key < slot[j-1].key)
        swap(slot[j], slot[j-1]);
    else break;
    }
}
```

sorted.

- Even if the search fails, the element of A[1...i-1] that is last checked by Binary search closest to ith element thereby indicating the correct posititon for ith element

- In the worst case, the number of key comparisons for the ith element is $\Theta \log_2 i$

→ for $i = 2, 3. \ldots n$, the total no of key comparisons in worst case is

$$\sum_{i=2}^{n} \Theta \log_2 i = O(n \log_2 n)$$