COMP90038 Algorithms and Complexity

Tutorial 3 Brute Force and Recursion

Tutor: Chuang(Frank) Wang

1. Review

Brute Force

Brute force:

A brute force algorithm simply tries all possibilities exhaustively until a satisfactory solution is found.

- Simple, easy to implement
- inefficient—does not scale well.

- **Sorting** - Selection Sort

  Select the 1st smallest item &, select the 2nd smallest item…select the nth smallest item(largest).

- **Optimizing** - Find the best solution
  - Require finding all solutions, or if a value for the best solution is known, it may stop when any best solution is found
  - e.g. Travelling Salesperson (TSP), Closest Pair

- **Satisficing** - Search for an item with a particular property
  - Stop as soon as a solution is found that is good enough
  - e.g. String matching

Sorting

- **in-place:** if it does not require additional memory for manipulating the input. *However, a small constant extra space used for variables is allowed.*
- **stable:** if it preserves the relative order of elements that have identical keys.
- **input-insensitive:** if its running time is fairly independent of input properties other than size.

Selection Sort

```
function SelSort(A[0.. n-1])
    for i ← 0 to n − 2 do
        min ← i
        for j ← i+1 to n-1 do
            if A[j] < A[min] then
                min ← j
        t ← A[i]
        A[i] ← A[min]
        A[min] ← t
```

swap

Complexity: $\theta(n^2)$

- **in-place**
- **stable** if "swap" if "push back"
- **input-insensitive**

4	2	3	4	1

4	2	3	4	1

i	min	0	1	2	3	4	5	6	7	8	9	10
		S	O	R	T	E	X	A	M	P	L	E
0	6	S	O	R	T	E	X	A	M	P	L	E
1	4	A	O	R	T	E	X	S	M	P	L	E
2	10	A	E	R	T	O	X	S	M	P	L	E
3	9	A	E	E	T	O	X	S	M	P	L	R
4	7	A	E	E	L	O	X	S	M	P	T	R
5	7	A	E	E	L	M	X	S	O	P	T	R
6	8	A	E	E	L	M	O	S	X	P	T	R
7	10	A	E	E	L	M	O	P	X	S	T	R
8	8	A	E	E	L	M	O	P	R	S	T	X
9	9	A	E	E	L	M	O	P	R	S	T	X
10	10	A	E	E	L	M	O	P	R	S	T	X
		A	E	E	L	M	O	P	R	S	T	X

entries in black are examined to find the minimum

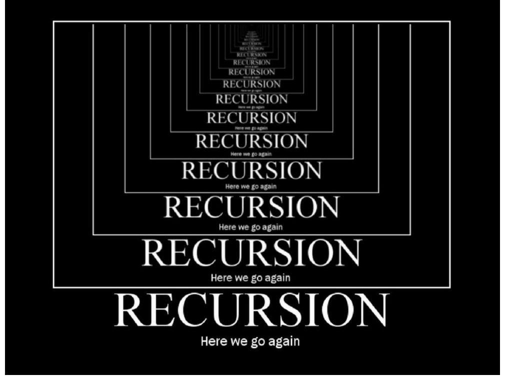
entries in red are a[min]

entries in gray are in final position

Trace of selection sort (array contents just after each exchange)

Sourced from Sedgewick, R., & Wayne, K. (2011). Algorithms. Addison-wesley professional.

Recursion



Recursion:

Recursion is a process by which a function **calls *itself* directly or indirectly**. The corresponding function is called as recursive function.

Two main components required for every recursive function.

- ***Base case:***
  - ✓ returns a value without making any subsequent recursive calls
  - ✓ The value returned in base case is provided.
  - ✓ The role of the base condition is to **stop a recursive function from executing endlessly**. This is when the function keeps calling itself… and never stops calling itself!

stackoverflow

- ***Reduction part:***
  - ✓ relates the value of the function at one (or more) input values to the value of the function at one (or more) other input values.
  - ✓ the sequence of input values must **converge** to the base case.

Factorial

$$n! = \begin{cases} n * (n-1)!, & n \geq 1 \\ 1, & n = 0 \end{cases}$$

```
function factorial(n)
    if n = 0 then
        return 1
    return factorial(n-1) * n
```

```
factorial(5)
   factorial(4)
      factorial(3)
         factorial(2)
            factorial(1)
               return 1
            return 2*1 = 2
         return 3*2 = 6
      return 4*6 = 24
   return 5*24 = 120
```

the function knows it's time to exit.

Iteration VS Recursion

	Iteration	Recursion
Definition	A set of instructions repeatedly executed.	Function calls itself.
Termination	the loop's termination condition is satisfied	the base case
Complexity	Relatively lower time complexity, e.g. • linear for a single loop • quadratic for a nested loop	Very high(generally exponential) time complexity.
Example	see iterative code below	see recursive code below

Iteration example:

```
function factorial(n)
    result ← 1
    while n > 0 do
        result ← result * n
        n ← n - 1
    return result
```

Recursion example:

```
function factorial(n)
    if n = 0 then
        return 1
    return factorial(n-1) * n
```

2. Tutorial Questions

THE UNIVERSITY OF MELBOURNE

Question 15

15. For each of the following pairs f, g, determine whether $f(n) \in O(g(n))$, or $g(n) \in O(f(n))$, or both:

(a) $f(n) = (n^2 + 1 - n^2)/2$ and $g(n) = 2n$
(b) $f(n) = n^2 + n\sqrt{n}$ and $g(n) = n^2 + n$
(c) $f(n) = n \log n$ and $g(n) = \frac{n}{4}\sqrt{n}$
(d) $f(n) = n + \log n$ and $g(n) = \sqrt{n}$
(e) $f(n) = 4n \log n + n$ and $g(n) = (n^2 - n)/2$
(f) $f(n) = (\log n)^2$ and $g(n) = 2 + \log n$

Solution:

L'Hôpital's Rule

$$\lim_{h\to\infty} \frac{t(n)}{g(n)} = \lim_{h\to\infty} \frac{t'(n)}{g'(n)}$$

where t' and g' are the derivatives of t and g

OR find the dominant term

(a) $\lim_{n\to\infty} \frac{1/2}{2n} = \lim_{n\to\infty} \frac{1}{4n} = 0$ $\qquad f(n) \in O(g(n))$

(b) $\lim_{n\to\infty} \frac{n^2+n^{\frac{3}{2}}}{n^2+n} = \lim_{n\to\infty} \frac{2n+\frac{3}{2}n^{\frac{1}{2}}}{2n+1} = \lim_{n\to\infty} \frac{2+\frac{3}{4}n^{-\frac{1}{2}}}{2} = constant$ $\qquad f(n) \in O(g(n))$ and $g(n) \in O(f(n))$

(c) $\lim_{n\to\infty} \frac{nlogn}{\frac{1}{4}n^{\frac{3}{2}}} = \lim_{n\to\infty} \frac{16}{3\sqrt{n}} = 0$ $\qquad f(n) \in O(g(n))$

(d) $\lim_{n\to\infty} \frac{n+logn}{\sqrt{n}} = \lim_{n\to\infty} \frac{4}{n^{-1/2}} = \infty$ $\qquad g(n) \in O(f(n))$

(e) $\lim_{n\to\infty} \frac{4nlogn+n}{(n^2-n)/2} = \lim_{n\to\infty} \frac{4}{n} = 0$ $\qquad f(n) \in O(g(n))$

(f) $\lim_{n\to\infty} \frac{(logn)^2}{2+logn} = \lim_{n\to\infty} 2logn = \infty$ $\qquad g(n) \in O(f(n))$

Question 16

16. Show the steps of selection sort, when given the keys S, O, R, T, X, A, M, P, L, E.

Solution:

i	min	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]
		S	O	R	T	E	X	A	M	P	L	E
0	6	S	O	R	T	E	X	A	M	P	L	E
1	4	A	O	R	T	E	X	S	M	P	L	E
2	10	A	E	R	T	O	X	S	M	P	L	E
3	9	A	E	E	T	O	X	S	M	P	L	R
4	7	A	E	E	L	O	X	S	M	P	T	R
5	7	A	E	E	L	M	X	S	O	P	T	R
6	8	A	E	E	L	M	O	S	X	P	T	R
7	10	A	E	E	L	M	O	P	X	S	T	R
8	8	A	E	E	L	M	O	P	R	S	T	X
9	9	A	E	E	L	M	O	P	R	S	T	X
10	10	A	E	E	L	M	O	P	R	S	T	X
		A	E	E	L	M	O	P	R	S	T	X

entries in black are examined to find the minimum

entries in red are a[min]

entries in gray are in final position

Trace of selection sort (array contents just after each exchange)

Sourced from Sedgewick, R., & Wayne, K. (2011). Algorithms. Addison-wesley professional.

Question 18

18. Trace the brute-force string search algorithm on the following input: The path p is ‘`needle`’, and the text t is ‘`there_need_not_be_any`’. How many comparisons (successful and unsuccessful) are made?

Solution:

```
function str_matching(p[0..m-1], t[0..n-1])
    for i ← 0 to n − m do
        j ← 0
        while j < m and p[j] = t[i + j] do
            j ← j + 1
        if j = m then
            return i
    return -1
```

t

t	h	e	r	e	_	n	e	e	d	_	n	o	t	_	b	e	_	a	n	y
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

p

n	e	e	d	l	e
0	1	2	3	4	5

$$1 + 1 + 1 + 1 + 1 + 1 + 5 + 1 + 1 + 1 + 1 + 2 + 1 + 1 + 1 + 1 = 21$$

Question 19

19. Assume we have a text consisting of one million zeros. For each of these patterns, determine how many character comparisons the brute-force string matching algorithm will make:

(a) `010001` (b) `000101` (c) `011101`

Solution:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	.	.	.	.	.	.	.	.	.	.	.	999,993	999,994	999,995	999,996	999,997	999,998	999,999

0	1	0	0	0	1
0	1	2	3	4	5

(a) 2×999995 comparisons

0	0	0	1	0	1
0	1	2	3	4	5

(b) 4×999995 comparisons

0	1	1	1	0	1
0	1	2	3	4	5

(c) 2×999995 comparisons

Question 20

20. Give an example of a text of length n and a pattern of length m, which together constitute a worst-case scenario for the brute-force string matching algorithm. How many character comparisons, as a function of n and m, will be made for the worst-case example. What is the value of m (the length of the pattern) that maximises this function? i.e. What is the worst case pattern length?

Solution:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	.	.	.	.	.	.	.	.	.	.	.	999,993	999,994	999,995	999,996	999,997	999,998	999,999

0	0	0	0	0	1
0	1	2	3	4	5

The worst case happens when

- Text(length n): consisting of the same character c repeated n times
- Pattern(length m): consisting of m − 1 occurrences of c, followed by a single character different from c.

In this case

- the outer loop is traversed n − m + 1 times
- each time, m character comparisons are made before failure is detected.
- Altogether we have $(n - m + 1)m = (n + 1)m - m^2$ comparisons.

- When $2m \approx n$, this has its maximal value where n + 1 − 2m = 0. (assume n is extremely large)

THE UNIVERSITY OF MELBOURNE

Question 21

21. The *assignment problem* asks how to best assign n jobs to n contractors who have put in bids for each job. An instance of this problem is an $n \times n$ *cost matrix* C, with $C[i,j]$ specifying what it will cost to have contractor i do job j. The aim is to minimise the total cost. More formally, we want to find a permutation $\langle j_1, j_2, \ldots j_n \rangle$ of $\langle 1, 2, \ldots, n \rangle$ such that $\sum_{i=1}^{n} C[i, j_i]$ is minimized. Use brute force to solve the following instance:

	Job 1	**Job 2**	**Job 3**	**Job 4**
Contractor 1	9	2	7	8
Contractor 2	6	4	3	7
Contractor 3	5	8	1	8
Contractor 4	7	6	9	4

Permutation	**Cost**
1,2,3,4	9+4+1+4 = 18
1,2,4,3	9+4+8+9 = 30
1,3,2,4	9+3+8+4 = 24
1,3,4,2	9+3+8+6 = 26
1,4,2,3	9+7+8+9 = 33
1,4,3,2	9+7+1+6 = 23
2,1,3,4	2+6+1+4 = 13
2,1,4,3	2+6+8+9 = 25
⋮	

The minimal cost is 13, for permutation ⟨ 2, 1, 3, 4 ⟩ .

Question 22

22. Give an instance of the assignment problem which does not include the smallest item $C[i, j]$ of its cost matrix.

	Job 1	Job 2
Contractor 1	1	2
Contractor 2	2	4

The minimal cost is 4, for permutation $\langle 2, 2 \rangle$ which doesn't include the smallest item 1.

Question 17

17. One possible way of representing a polynomial

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

is as an array A of length $n+1$, with $A[i]$ holding the coefficient a_i.

(a) Design a brute-force algorithm for computing the value of $p(x)$ at a given point x. Express this as a function PEVAL(A, n, x) where A is the array of coefficients, n is the degree of the polynomial, and x is the point for which we want the value of p.

(b) If your algorithm is $\Theta(n^2)$, try to find a linear algorithm.

(c) Is it possible to find an algorithm that solves the problem in sub-linear time?

```
function peval(A, n, x)
    result ← 0.0
    for i ← n downto 0 do
        summand ← 1.0
        for j ← 1 to i do
            summand ← x * summand
        result ← result + A[i] * summand
    return result
```

Complexity: $O(n^2)$

```
function peval(A, n, x)
    result ← A[n]
    for i ← n-1 downto 0 do
        result ← result*x + A[i]
    return result
```

$$7 * x^4 + 3 * x^3 + 4 * x^2 - x - 5$$
$$= x * (x * (x * (7x + 3) + 4) - 1) - 5$$

Complexity: $O(n)$

(C)

We cannot solve the problem in less than linear time, because we clearly need to access each of the n + 1 coefficients.

THE UNIVERSITY OF
MELBOURNE

Next Week:

Graphs, trees, graph traversal and allied algorithms(BFS, DFS).

Thank you !