**CSE102 Algorithmic Foundations**

1. the order of the big O:

O(1), O(log n), O(log2 n), O(), O(n), O(n log n), O(n3/2), O(n2), O(n2 log4 n), O(n3), O(n10/3 ), O(n4).

2. Binary Search: O(log n)

Input should be a sequence of n **sorted** numbers.

RecurBinarySearch(A, first, last, X)

begin

if (first > last) then

return false

mid = ⎣(first + last)/2⎦

if (X == A[mid]) then

return true

if (X < A[mid]) then

return RecurBinarySearch(A, first, mid-1, X)

else

return RecurBinarySearch(A, mid+1, last, X)

end

3. Merge sort: O (log n)

Algorithm Mergesort(A[0..n-1])

if n > 1 then begin

copy A[0..⎣n/2⎦-1] to B[0..⎣n/2⎦-1]

copy A[⎣n/2⎦..n-1] to C[0..⎡n/2⎤-1]

Mergesort(B[0..⎣n/2⎦-1])

Mergesort(C[0..⎡n/2⎤-1])

Merge(B, C, A)

end

Algorithm Merge(B[0..p-1], C[0..q-1], A[0..p+q-1])

Set i=0, j=0, k=0

while i<p and j<q do

begin

if B[i]≤C[j] then set A[k]=B[i] and increase i

else set A[k] = C[j] and increase j

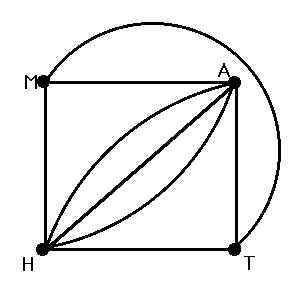
k = k+1

end

if i==p then copy C[j..q-1] to A[k..p+q-1]

else copy B[i..p-1] to A[k..p+q-1]

4. Vertex & Degree



|  |  |
| --- | --- |
| Vertex | Degree |
| M | 3 |
| A | 5 |
| T | 3 |
| H | 5 |

*d*

*b*

*c*

*a*

*h*

*g*

*f*

*e*

5. Adjacency matrix / list

adjacency matrix



adjacency list

a

e

b

b

a

f

c

b

d

g

c

d

c

e

f

a

b

g

f

g

h

e

c

h

f

g

6. Incidence matrix / list

Let *e1=(a, b), e2= (a, e), e3= (b, c) e4= (b, f), e5= (c, g), e6= (c, d), e7= (e, f), e8= (f, g), e9= (g, h)*

incidence matrix



incidence list

**e1**

**b**

**a**

**e2**

**e**

**a**

**e3**

**c**

**b**

**e4**

**f**

**b**

**e5**

**g**

**c**

**e6**

**d**

**c**

**e7**

**f**

**e**

**e8**

**g**

**f**

**e9**

**h**

**g**

7. Euler path:

An Euler path is a path visiting every edge of G exactly once.

8. Euler circuit:

An Euler circuit in a graph G is a circuit visiting every edge of G exactly once.  
(NB. A vertex can be repeated.)

|  |  |  |
| --- | --- | --- |
|  | Exist Euler circuit? | Exist Euler path? |
| all vertices have even degree | YES | YES |
| exactly two vertices have odd degree | NO | YES |
| more than 2 vertices have odd degree | NO | NO |

9. BFS & DFS

Consider the following graph G.

*d*

*b*

*c*

*a*

*h*

*g*

*f*

*e*

DFS:

*a*

*b*

*c*

*d*

*e*

*f*

*g*

*h*

*a*

*b*

*c*

*d*

*e*

*f*

*g*

*h*

BFS:

10. preorder traversal

Start from root, the same order as DFS.

11. inorder traversal

Start from left most node, the root will be traversed if all the left sub-tree nodes are traversed.

12. postorder traversal

Start from left most node, the root will be traversed if all the left and right sub-tree nodes are traversed.

13. Minimum Spanning tree (MST):

a spanning tree of G with minimum weight

14. Prim’s algorithm

15. Kruskal’s algorithm

16. Dijkstra’s algorithm

17. Assembly line scheduling with DP

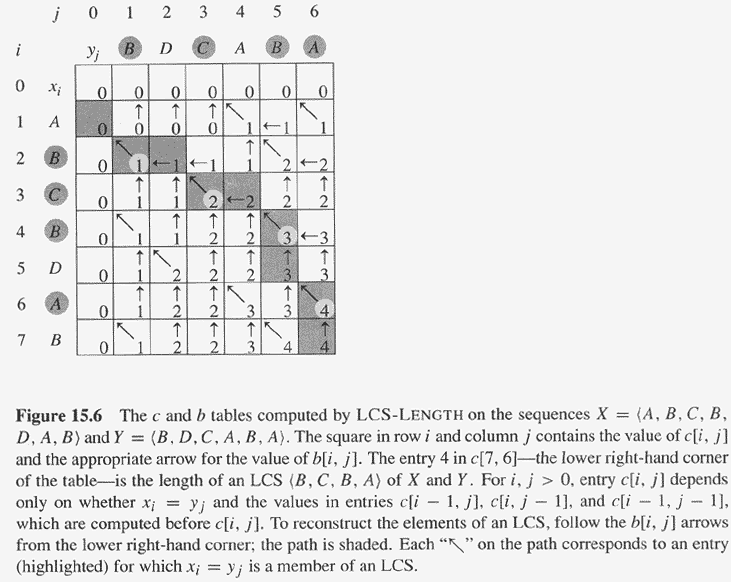
18. Counting sort

19. Horspool’s Algorithm

|  |  |
| --- | --- |
| 模式的长度*m* | 如果c不包含在模式的前m-1个字符中 |
| 模式前*m*−1个字符中最右边的*c*到模式最后一个字符的距离(不包括该字符) | 其他情况下 |

20. Bellman-Ford algorithm: Double Dijkstra’s algorithm.

21. LCS computation



22. Global vs. Local alignment

23. P NP NPC

24. Branch-and-Bound

lb = Σ (distances of two nearest cities of v in G)/2

25. Knapsack problem

26. Nearest-Neighbor Algorithm for TSP

TSP: Twice Around the Tree