PPT上的代码

2025年3月6日 星期四 21:21

插入排序 $\Theta(n^2)$

INSERTION-SORT(A)	cost	times
for $j = 2$ to $A.length$	c_1	n
$key = A[j]$	c_2	$n-1$
// Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
$i = j - 1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n}(t_j - 1)$
$i = i - 1$	c_7	$\sum_{j=2}^{n}(t_j - 1)$
$A[i+1] = key$	c_8	$n-1$

归并排序 $\Theta(n\lg n)$

Sorting Problem – Merge Sort

MSMA

■ **Pseudocode**

```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q+1, r)
    MERGE(A, p, q, r)
```

```
MERGE(A, p, q, r)
n1 = q - p + 1
n2 = r - q
let L[1 .. n1 + 1] and R[1 .. n2 + 1] be new arrays
for i = 1 to n1
    L[i] = A[p + i - 1]
for j = 1 to n2
    R[j] = A[q + j]
L[n1 + 1] = ∞
R[n2 + 1] = ∞
i = 1
j = 1
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```

最大子数组 $\Theta(n\lg n)$

Maximum-Subarray Problem

MSMA

■ **Time Complexity** Def. $T(n)$= The running time on any instance of size n.

$$T(n) = \Theta(n\lg n)$$

```
FIND-MAXIMUM-SUBARRAY(A, low, high)
if high == low
    return (low, high, A[low])      ← Θ(1)
else mid = ⌊(low + high)/2⌋
    (left-low, left-high, left-sum) = FIND-MAXIMUM-SUBARRAY(A, low, mid)
    (right-low, right-high, right-sum) = FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)
    (cross-low, cross-high, cross-sum) = FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
    if left-sum ≥ right-sum and left-sum ≥ cross-sum
        return (left-low, left-high, left-sum)
    elseif right-sum ≥ left-sum and right-sum ≥ cross-sum
        return (right-low, right-high, right-sum)
    else return (cross-low, cross-high, cross-sum)
```

← $2T(n/2)$

← $\Theta(n)$

← $\Theta(1)$

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$$

■ **A solution using divide and conquer**

$\Theta(n)$ = Find a max *prefix* + Find a max *suffix*

```
FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
left-sum = −∞
sum = 0
for i = mid downto low
    sum = sum + A[i]
    if sum > left-sum
        left-sum = sum
        max-left = i
right-sum = −∞
sum = 0
for j = mid + 1 to high
    sum = sum + A[j]
    if sum > right-sum
        right-sum = sum
        max-right = j
return (max-left, max-right, left-sum + right-sum)
```

最大子数组 - 改进

Maximum-Subarray Problem

MSMA

■ **Another divide and conquer algorithm – Time Complexity**

```
FIND-MAXIMUM-SUBARRAY-FAST-DIVE(A, l, r)
if l = r then
    return (l, r, A[l], A[l], A[l])      ← Θ(1)
else if l = r − 1 then
    return (l, r, A[l], A[l], A[l])      ← Θ(1)
else
    mid = ⌊(l + r)/2⌋
    (l1, r1, m1, p1, s1) = FIND-MAXIMUM-SUBARRAY-FAST-DIVE(A, l, mid)
    (l2, r2, m2, p2, s2) = FIND-MAXIMUM-SUBARRAY-FAST-DIVE(A, mid + 1, r)
    cross-subarray-sum = max(s1 + p2, m1, m2)
    prefix = max(p1, l1 + p2)
    suffix = max(s2, r2 + s1)
    total-sum = l1 + l2
    return (l1, r2, cross-subarray-sum, prefix, suffix, total-sum)
end if
```

← $2T(n/2)$

← $\Theta(1)$

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = n = 1 \\ 2T(n/2) + \Theta(1) & \text{if } n > 1 \end{cases}$$

算法	时间复杂度
插入排序	$\Theta(n^2)$
归并排序	$\Theta(n\lg n)$
最大子数组	$\Theta(n\lg n)$
钢条切割	$\Theta(n^2)$
矩阵链乘	$\Theta(n^3)$
LCS	$\Theta(mn)$
活动选择	$\Theta(n\lg n)$
Bellman-Ford	$O(VE)$
Dijkstra	$O(V^2)$
最短路径矩阵乘法	$O(n^3\lg n)$
Floyd	$\Theta(n^3)$
有向无环图	$\Theta(V+E)$
Ford-Fulkerson	$O(E\lvert f^*\rvert)$
Edmonds-Karp	$O(VE^2)$
MakeSpan	$O(n\lg m)$
顶点覆盖	$O(V+E)$
旅行商问题	$O(E\lg V)$

钢条切割的 $\Theta(n^2)$

Rod-Cutting Problem

MMA

■ Top-down with Memoization（备忘）- pseudocode

```
MEMOIZED-CUT-ROD(p, n)
let r[0..n] be a new array        An auxiliary array to save results of subproblems
for i = 0 to n
    r[i] = −∞
return MEMOIZED-CUT-ROD-AUX(p, n, r)

MEMOIZED-CUT-ROD-AUX(p, n, r)
if r[n] ≥ 0
    return r[n]
if n == 0
    q = 0
else q = −∞
    for i = 1 to n
        q = max(q, p[i] + MEMOIZED-CUT-ROD-AUX(p, n − i, r))
r[n] = q
return q
```

```
CUT-ROD(p, n)
if n == 0
    return 0
q = −∞
for i = 1 to n
    q = max(q, p[i] + CUT-ROD(p, n − i))
return q
```

Rod-Cutting Problem

■ Bottom-up – pseudocode

```
BOTTOM-UP-CUT-ROD(p, n)
let r[0..n] be a new array
r[0] = 0
for j = 1 to n
    q = −∞
    for i = 1 to j
        q = max(q, p[i] + r[j − i])      切 or 不切
    r[j] = q
return r[n]
```

Matrix-Chain Multiplication Problem $\Theta(n^3)$

■ 3. Computing the optimal costs - bottom-up - psedocode

$$m[i,j] = \begin{cases} 0 & \text{if } i = j, \\ \min_{i \le k < j} \{m[i,k] + m[k+1,j] + p_{i-1}p_kp_j\} & \text{if } i < j. \end{cases}$$

遍历长度为这段序列长度量为

```
MATRIX-CHAIN-ORDER(p)
n = p.length − 1
let m[1 .. n, 1 .. n] and s[1 .. n − 1, 2 .. n] be new tables
for i = 1 to n
    m[i, i] = 0                          if l is the chain length
for l = 2 to n
    for i = 1 to n − l + 1
        j = i + l − 1
        m[i, j] = ∞
        for k = i to j − 1               k是结合位置
            q = m[i, k] + m[k + 1, j] + p_{i−1}p_kp_j
            if q < m[i, j]
                m[i, j] = q
                s[i, j] = k
return m and s
```

Theorem 15.1 (Optimal substructure of an LCS)
Let $X = \langle x_1, x_2, \ldots, x_m \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$ be sequences, and let $Z = \langle z_1, z_2, \ldots, z_k \rangle$ be any LCS of X and Y.

1. If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} is an LCS of X_{m-1} and Y_{n-1}.
2. If $x_m \neq y_n$, then $z_k \neq x_m$ implies that Z is an LCS of X_{m-1} and Y.
3. If $x_m \neq y_n$, then $z_k \neq y_n$ implies that Z is an LCS of X and Y_{n-1}.

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ c[i-1,j-1] + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j, \\ \max(c[i,j-1], c[i-1,j]) & \text{if } i, j > 0 \text{ and } x_i \neq y_j. \end{cases}$$

$\Theta(mn)$

Longest Common Subsequence problem

■ Step 3: Computing the length of an LCS

```
LCS-LENGTH(X, Y)
m = X.length
n = Y.length
let b[1 .. m, 1 .. n] and c[0 .. m, 0 .. n] be new tables
for i = 1 to m
    c[i, 0] = 0
for j = 0 to n
    c[0, j] = 0
for i = 1 to m
    for j = 1 to n
        if x_i == y_j
            c[i, j] = c[i − 1, j − 1] + 1
            b[i, j] = "↖"
        elseif c[i − 1, j] ≥ c[i, j − 1]
            c[i, j] = c[i − 1, j]
            b[i, j] = "↑"
        else c[i, j] = c[i, j − 1]
            b[i, j] = "←"
return c and b
```

$X = \langle A, B, C, B, D, A, B \rangle$
$Y = \langle B, D, C, A, B, A \rangle$

		B	D	C	A	B	A
	0	0	0	0	0	0	0
A	0	0	0	0	1	1	1
B	0	1	1	1	1	2	2
C	0	1	1	2	2	2	2
B	0	1	1	2	2	3	3
D	0	1	2	2	2	3	3
A	0	1	2	2	3	3	4
B	0	1	2	2	3	4	4

```
GREEDY-ACTIVITY-SELECTOR(s, f)
n = s.length
SORT activities by finish times and renumber so that f_1 ≤ f_2 ≤ ··· ≤ f_n
A = {a_1}
k = 1
for m = 2 to n do
    if s[m] ≥ f[k] then
        A = A ∪ {a_m}
        k = m
    end if
end for
return A
```

DIJKSTRA(G, w, s) $O(V^2)$

```
INITIALIZE-SINGLE-SOURCE(G, s)
S=∅                    → 已找到最短路的点集合
Q=G.V                  → 未找到 - - -
while Q≠∅
    u=EXTRACT-MIN(Q)
    S=S∪{u}
    for each vertex v∈G.Adj[u]
        RELAX(u, v, w)
```

```
INIT-SINGLE-SOURCE(G, s)
for each v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```

Bellman-Ford 算法 $O(VE)$

```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i=1to |G.V| − 1
    for each edge(u, v)∈G.E
        RELAX(u, v, w)
for each edge(u, v)∈G.E
    if v.d>u.d+w(u, v)
        return FALSE
return TRUE
```

```
RELAX(u, v, w)
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

最短路径和矩阵乘法 $\Theta(n^3\lg n)$

```
EXTEND-SHORTEST-PATHS(L, W)
n=L.rows
let L'(l'_ij)be a new n×n matrix
for i=1 to n
    for j=1 to n
        l'_ij=∞
        for k=1 to n
            l'_ij=min(l'_ij, l_ik+w_kj)
return L'
```

```
SLOW-ALL-PAIRS-SHORTEST-PATHS(W)
n=W.rows
L^(1)=W
for m=2 to n−1
    let L^(m) be a new n×n matrix
    L^(m)=EXTEND-SHORTEST-PATHS(L^(m−1), W)
return L^(n−1)
```

```
FASTER-ALL-PAIRS-SHORTEST-PATHS(W)
n=W.rows
L^(1)=W
m=1
while m<n−1
    let L^(2m) be a new n×n matrix
    L^(2m)=EXTEND-SHORTEST-PATHS(L^(m), L^(m))
    m=2m
return L^(m)
```

Floyd-Warshall 算法 $\Theta(n^3)$ → 邻接矩阵

```
FLOYD-WARSHALL(W)
n=W.rows
D^(0)=W
for k=1 to n
    let D^(k)=(d_ij^(k))be a new n×n matrix
    for i=1 to n
        for j=1 to n
            d_ij^(k)=min(d_ij^(k−1), d_ik^(k−1)+d_kj^(k−1))
return D^(n)
```

有向无环图单源最短路径 $\Theta(V+E)$

```
DAG-SHORTEST-PATHS(G, w, s)
topologically sort the vertices of G
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex u, taken in topologically sorted order
    for each vertex v∈G.Adj[u]
        RELAX(u, v, w)
```

FORD-FULKERSON(G, s, t) $O(E\lvert f^*\rvert)$ → 最大流 流量

```
FORD-FULKERSON(G, s, t)
for each edge(u, v)∈G.E
    (u, v). f=0
while there exists a path p from s to t in the residual network G_f
    c_f(p)=min{c_f(u, v):(u, v)∈p}
    for each edge(u, v)∈p
        if(u, v)∈E
            (u, v). f=(u, v). f+c_f(p)
        else(v, u). f=(v, u). f−c_f(p)
```

Edmonds-Karp $O(VE^2)$

上3

GREEDYMAKESPAN algorithm $O(n\log m)$

```
for i = 1 to m do
    T_i ← 0;                          ▷ T_i denotes the load of machine i
    A_i ← ∅;                          ▷ initialize all machines with 0 jobs
end for
for j = 1 to n do
    Find a k such that T_k = min_{1≤i≤m} T_i;
    A_k ← A_k ∪{j};                   ▷ assign job j to machine M_k
    T_k ← T_k + t_j;
end for
```

APPROX-VERTEX-COVER(G) $O(V+E)$

```
C = ∅
E' = G.E
while E' ≠ ∅
    let (u, v) be an arbitrary edge of E'
    C = C ∪ {u, v}
    remove from E' every edge incident on either u or v
return C
```

$O(V + E)$, using adjacency lists to represent E'

MST-PRIM(G, w, r) $O(E\lg V)$

```
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```

```
APPROX-TSP-TOUR(G, c)
select a vertex r ∈ G.V to be a "root" vertex
compute a minimum spanning tree T for G from root r
    using MST-PRIM(G, c, r)
let H be a list of vertices, ordered according to when they are first visited
    in a preorder tree walk of T
return the hamiltonian cycle H
```