Expriment 3

PB17111623 范睿

2019 年 11 月 25 日

1 偏序关系

1.1 题目

对于二维平面上的任意两点$A(x_1, y_1), B(x_2, y_2)$定义偏序关系$\leq$，表示$x_1 \leq x_2$ 且$y_1 \leq y_2$ 。现给定平面上的若干点，求最大的点的子集S使得集合中的任意两点之间均满足偏序关系$\leq$，即对$\forall A, B \in S, A \leq B$或$B \leq A$成立。只需要输出子集S的大小$|S|$。

输入:
第一行一个整数n表示点的个数。然后是n行输入，表示n个点的坐标，其中每行的格式为：$x_i \quad y_i$点的横纵坐标均为整数，且输入数据不会出现重叠的点。

数据规模：
$0 < n \leq 10000$ $0 < x_i, y_i \leq 10000$

输出：
输出一个整数表示最大可能的子集$|S|$的大小。

Sample Input:
5
0 1
1 4
2 5
3 3
4 2

Sample Output:
3
/* (0,1), (1,4), (2,5)

*/

1.2 思路

寻找问题子结构：第i个坐标所属的拥有最大个数的满足偏序关系的集合=第i个坐标+前i-1个坐标所属的最大集合（前提是此合并后的集合满足偏序关系）
因此先将所有坐标从小到大排序，比较规则是：先看x分量，x分量大的坐标大，若x相同，则看y分量，y分量大的坐标大。
然后设一个长度为n的数组A[1...n]。A[i]记录：在前i个坐标中，包含第i个坐标的最大集合大小$A[i].num$和该集合的最大与最小元素$A[i].S$。
得到递归式：

$$A[i] = \max_{1\leq j\leq i}(|A[j].S + (x_i, y_i)|)$$

其中$|A[j].S + (x_i, y_i)|$的计算如下：
若(x_i, y_i)按照偏序比较规则大于等于$A[j].S$中的最大的或小于等于最小的，那么$|A[j].S + (x_i, y_i)| = |A[j].S| + 1$。
否则说明(x_i, y_i)与$A[j].S$无法形成偏序关系，那么$|A[j].S+(x_i, y_i)| = 1$，即集合中只有(x_i, y_i)一个。

当找到最大值对应的j时，在更新$A[i].num$的同时，还需要更新$A[i].S$，即要将$A[i].S$换成$A[j].S$和(x_i, y_i)组成偏序关系后的集合。
当i=1时，$A[i].num = 1, A[i].S.(x_{min}, y_{min}) = (x_i, y_i), A[i].S.(x_{max}, y_{max}) = (x_i, y_i)$
最终答案为所有num中最大的那个。

排序的原因：
若先排了序，那么第A[i].num至少为2（i不为1时），即(x_i, y_i)至少可以找到一个j的集合与它组成偏序关系。

1.3 算法

1.4 复杂度分析

排序利用冒泡排序，时间为$\mathcal{O}(n^2)$，遍历i从1到n，对每个i来说又有j从i-1到1，时间为$\mathcal{O}(n^2)$，因此总时间为$\mathcal{O}(n^2)$ 。

Algorithm 1 Poset

输入: 坐标个数n，每个坐标值(x_i, y_i) $1 \leq i \leq n$

输出: 最大偏序集合大小$|S|$

```
Sort (x_i, y_i, 1 ≤ i ≤ n)
let A[1...n] be a new array
max ← 0
for i = 1 to n do
    A[i].num = 1
    A[i].S.x_min = x_i
    A[i].S.y_min = y_i
    A[i].S.x_max = x_i
    A[i].S.y_max = y_i
    for j = i-1 downto 1 do
        if {(x_i, y_i)} ∪ A[j].S is a Partiallyordered set then
            if A[j].num ≥ A[i].num then
                A[i].S = {(x_i, y_i)} ∪ A[j].S
                A[i].num ← A[j].num + 1
            end if
        end if
    end for
    if A[i].num ≥ sum then
        max ← A[i].num
    end if
end for
return max
```

2 归并问题

2.1 题目

程序员小明需要将nn个有序数组合并到一起，由于某种工程上的原因，小明只能使用一个系统函数Merge将两个相邻的数组合并为一个数组。Merge函数合并两个长度分别为n_1, n_2的数组的时间为$n_1 + n_2$。现给定nn个数组的长度，小明需要求出最小需要的合并时间。

输入：
输入共两行。第一行一个整数n表示待归并的数组个数。第二行n个整数，第i个整数表示第i个数组的长度。

数据规模：
$0 < n \leq 200$ 每个数组的长度均为整数，且输入数据保证最终结果范围在int32之内。

输出：
输出共一个数字，表示最小需要的归并时间。

Sample Input:
5
68 34 41 55 71

Sample Output:
613

2.2 思路

寻找子问题结构：
与矩阵链相乘相似，若求第i到j个元素相乘的最小合并时间，将k从i遍历到j-1，用k将i...j分割成两个子序列。那么i到j元素相乘的最小合并时间转换成求得一个在k处的分割，使得i...k和k+1...j的合并时间之和加i到j所有元素相加的总值最小。
那么设一个二维数组A[1...n][1...n],A[i][j]存储i...j的元素相乘的最小合并时间。那么有递归式：

$$A[i][j] = \min_{i \leq k < j}(A[i][k] + A[k+1][j] + \sum_{p=i}^{j} n_p)$$

边界条件为：$A[i][i] = 0, 1 \leq i \leq n$

Algorithm 2

输入: 每个数组长度$n_i, 1 \le i \le n$, 数组个数n

输出: mintime

```
let A[1...n][1...n] be a new array
for i = 1 to n do
    A[i][i] ← 0
end for
for l = 2 to n do
    for i = 1 to n-l+1 do
        j ← i + l − 1
        A[i][j] ← Inf
        s ← Σ_{p=i}^{j} n_p
        for k = i to j-1 do
            q ← A[i][k] + A[k + 1][j] + s
            if q < A[i][j] then
                A[i][j] ← q
            end if
        end for
    end for
end for
return A[1][n]
```

2.3 算法

2.4 复杂度分析

三层嵌套循环，时间复杂度为$\mathcal{O}(n^3)$

3 多重背包

3.1 题目

现有一个背包可以容纳总重量为W的物品，有n种物品可以放入背包，其中每种物品单个重量为w_i，价值为v_i，可选数量为num_i。输出可以放入背包的物品的最大总价值。

输入：
第一行两个整数n,W，分别表示物品件数和背包容量。然后n行数据描述每种物品的重量、价值和可选数量。每行的格式为 w_i v_i 。

数据规模：
$1 \leq n \leq 200, 1 \leq W \leq 10000, 1 \leq w_i \leq 1000, \leq v_i \leq 1000, \leq num_i \leq 10000$
所有输入数据均为整数。

输出:
输出一个整数表示可以装入背包的最大价值。

Sample Input:
5 100
7 3 68
10 3 161
10 6 55
5 2 14
3 10 165

Sample Output:
330

3.2 思路

寻找子文体结构：
若想求得空间为w时，存放第i到第n个物品的最优解，可以一步步尝试：
第i个物品放0个，value最大值=w空间放第i+1到第n个物品;第i个物品放1个，value最大值=(w-w_i)空间放第i+1到第n个物品的value最大值+1*v_i;...第i个物品放k个(k< num_i),value最大值==(w-$k*w_i$)空间放第i+1到第n个物品的value最大值+k*v_i;...第i个物品放k个(k≥ num_i),value最大

值==(w-$num_i * w_i$)空间放第i+1到第n个物品的value最大值+num_i*v_i
那么可以建立一个二维数组KS[0...W][1...n+1]，KS[i][j]表示当空间是i时，放第j到第n个物品的value最大值。有递归式：

$$KS[i][j] = \max_{0 \le k \le \lfloor \frac{j}{w_i} \rfloor} (KS[i - \max(num_i, k) * w_i][j+1] + \max(num_i, k) * v_i)$$

边界条件为：
KS[i][n+1] = 0,虚拟一个数量为0，价值为0的物品，序号为n+1，那么在KS表中，它对应的那一列均为0。

3.3 算法

Algorithm 3

输入: 物品个数n，背包大小W，每个物品的大小，价值，个数w_i, v_i, num_i
输出: 最大价值valuemax

let KS[0...W][1...n+1] be a new array
for i = 0 to W **do**
  $KS[i][n+1] \leftarrow 0$
end for
for i = n downto 1 **do**
  for j = 0 to W **do**
    $KS[j][i] \leftarrow 0$
    for $k = 0$ to $\lfloor \frac{j}{w_i} \rfloor$ **do**
      if $k > num_i$ **then**
        $value \leftarrow v_i * num_i + KS[j - num_i * w_i][i+1]$
      else
        $value \leftarrow v_i * k + KS[j - k * w_i][i+1]$
      end if
      if $value > KS[j][i]$ **then**
        $KS[j][i] \leftarrow value$
      end if
    end for
  end for
end for
return KS[W][1]

3.4 复杂度分析

$\mathcal{O}(W^2 * n)$

4 正方形计数

4.1 题目

现有一个$n \times m$的矩形区域，其中每个单位区域可能有损坏。要求找到地面上所有不包含损坏区域的正方形的个数。

输入:
第一行两个整数n,m表示矩形区域的大小。接下来共有n行输入数据，每行包含m个0或1的整数，其中0表示该地面完好，1表示该地面已损坏。

数据规模：
$0 < n, m \leq 2000$

输出:
输出一个整数表示区域内的正方形个数。输入数据保证结果不会超出int32的范围。

Sample Input:
5 5
1 0 0 0 0
1 0 0 1 1
0 1 0 0 0
0 1 0 1 0
1 0 0 0 0

Sample Output:
18

4.2 思路

寻找问题子结构：
从左上角开始，从所往右，从上往下遍历：若当前位置是1（未损坏），则当前位置的正方形个数是0；若为0（损坏），则当前位置正方形个数为其左边、上边、左上角的三个位置的正方形个数的最小值+1，于是定义二维数组A[0...n+1][0...m+1]有递归式：

$$A[i][j] = \min(A[i-1][j], A[i][j-1], A[i-1][j-1]) + 1, i, j \neq 0$$

边界条件为：
在给定的n*m的矩阵外部加一圈1，构成(n+1)*(m+1)的矩阵，那么二维数组A的最外围一

圈就是0
最终结果就是所有位置的正方形个数的和。

4.3 算法

Algorithm 4 dangerous goods

输入: 行数n，列数m，矩形区域M[1...n][1...m]
输出: 区域内正方形数sum

```
let A[0...n+1][0...m+1] be a new array
for i = 0 to n+1 do
    A[i][0] ← 0
    A[i][m+1] ← 0
end for
for i = 0 to m+1 do
    A[0][i] ← 0
    A[n+1][i] ← 0
end for
sum ← 0
for i = 1 to n do
    for j = 1 to m do
        if M[i][j] is 1 then
            A[i][j] ← 0
        else
            A[i][j] ← min(A[i−1][j], A[i][j], A[i−1][j−1]) + 1
            sum ← sum + A[i][j]
        end if
    end for
end for
return sum
```

4.4 复杂度分析

$\mathcal{O}(n \times m)$