091M4041H - Assignment 5
Algorithm Design and Analysis

Song Qige 2017E8018661044

2018 年 1 月 10 日

1 负载均衡(1)

1.1 algorithm describe and pseudo-code

自然语言：设有m个电脑n个工作。建立图，设定起点s和终点t，每个工作为一个点，将s和每个工作点连接一条边，流值设定为1。每个电脑为一个点，将每个工作和其可以选择的电脑连接一条边，流值设定为1。将每个电脑和t连接一条边，对于m 个电脑n个工作，极端情况是所有工作都在同一个电脑上进行，因此每个电脑到t连接的边的流值的范围为0到m。要求出最小的最大负载，即求出一个方案使每个电脑到t连接的所有边中的最大流值最小。此时采用二分查找实现。设定初始的l=0，r=m，mid=(l+r)/2，将mid的值作为每个电脑到t连接的所有边的流值，用ford-folkson算法对图求最大流，如果最大流的值等于工作数m，表示可以流通，则缩小范围为l=mid，mid=(l+r)/2。如果不能流通，则r=mid+1。继续将mid的值作为每个电脑到t连接的所有边的流值并对图求最大流，直到l<r不成立。最终的l即为最小的最大电脑负载值。

```
function CreateGraph(Graph, mid)
    for i = 1 → n do
        Graph.Addedge(s, job[i], 1)
    end for
    for i = 1 → n do
        for j = 1 → m do
            Graph.Addedge(job[i], computer[j], 1)
        end for
    end for
    for j = 1 → m do
        Graph.Addedge(computer[j], t, mid)
    end for
end function
function LoadBalance(G, m, n)
    l = 0
    r = m
    while l < r do
        mid = (l + r)/2
        createGraph(G, mid)
        maxflowvalue = G.MaxFlow('s', 't')
        if maxflowvalue == n then
            r = mid
        else
            l = mid + 1
        end if
    end while
    return l
end function
```

1.2 Prove the correctness

对于n个job，m个computer，当n个job全部在同一台computer上执行时，max load为n。题目需要找到max load最小的分配job的方案。对于上述建图方式，最终需要决定每个computer到终点t的流值，也就是在当前要求下，每个电脑上job的分配方案，其范围是0到m。
用二分的方式缩小这个流值的范围，设定其初始范围为[0,m]，mid=m/2。将mid作为computer到终点t的流值，如果求出图的最大流为job 数n，表明当前图所表示的分配方案可以完成全部工作。此时可将computer到终点t的流值的上界r更新为mid，重新计算mid，将其赋给computer到终点t的流值并计算最大流。
如果求出的最大流不等于job数n，说明当前分配方案不能完成所有job，此时应当增大computer到终点t的流值，将其范围的下界l 置为mid+1，重新计算mid，将其赋给computer到终点t的流值并计算最大流。
以上过程循环至$l \leq r$终止，得到的即为max load最小的分配job的方案，最小的max load值即为当前的l，表明将computer到终点t 的流值设为大于等于l的值，就一定能完成所有的job。

1.3 time complexity

建图：n+m+m*n次addedge操作。ford-folkson法求最大流，时间为复杂度O(mC)，其中m是边数，C是源点s出去的边的容量之和。本题中边数为n+m+n*m，C=n*1，即job 数。经过二分查找，有logm次求最大流，m是电脑数。所以$T(n) = O(logm * (n + m + n * m) * n)$。

2 求解符合条件的矩阵(2)

2.1 Algorithm Description and pseudo-code

自然语言：设矩阵有R行C列，共有R*C个元素，每行的元素之和对应数组a，每列的元素之和对应数组b。设源点s，终点t，设定R个代表行的点，将s点与这R个点连为边，容量设为a[i]。设定C个代表列的点，将这C个点

与t连为边，容量设为b[i]。将R个代表行的点都与C个代表列的点连接，形成R*C条边，代表矩阵中的R*C个元素，因为矩阵中元素的值为0或1，因此这R*C条边的容量范围是0到1。将其初值设为1，对构建的图求最大流，输出最终中间R*C条边的流值，即为符合条件的矩阵结果。

```
function MATRIX(Graph)
    for i = 1 → R do
        Graph.Addedge(s, i, a[i])
    end for
    for j = 1 → C do
        Graph.Addedge(R + j, t, b[i])
    end for
    for i = 1 → R do
        for j = 1 → C do
            Graph.Addedge(i, j + R, 1)
        end for
    end for
    Graph.maxflow()
    for i = 1 → R do
        for j = 1 → C do
            ans[i][j] = Graph.edge[i− > j + R].flow
        end for
    end for
    return ans
end function
```

2.2 Prove the correctness

对于以上建图方式求出最大流，则从源点s流向的R个点的流值代表每行的矩阵元素之和，然后流量分布到后续的R*C条边，每条边流值的上限

为1（流值为整数），会和到C个点，从C个点流向终点t的边的流值代表每列的矩阵元素之和。因此这样建图后求出最大流后中间R*C条边的流值即为符合题目要求的矩阵元素值，满足每行元素的和和每列元素的和等于给定值并且矩阵元素为0 或1。

2.3 time complexity

建图时间：R+C+R*C 次addedge操作。push-relabel求最大流时间:$O(n^2 * m)$，本题图n=R+C+2，m=R+C+R*C，所以最大流时间为$O((R+C+2)^2 * (R+C+R*C))$。所以$T(n) = O((R+C+2)^2 * (R+C+R*C))$。

3 唯一的最小割(3)

方法一：

3.1 Algorithm Description and pseudo-code

自然语言：用ford-folkson算法计算出G的最小割(S,T)，和对应的最大流F。对每条边e=(u,v)，u∈S,v∈T，增加C_e，重复ford-folkson算法，得到最大流F’。如果每个F’都比F大，则G有唯一的最小割，即最小割(S,T)。

function UNIQUECUT($Graph$)
    $(F, E_{cut}) = ford - folkson(Graph)$
    for $einE_{cut}$ **do**
        $C_e = C_e + 1$
        $(F', E'_{cut}) = ford - folkson(Graph)$
        if $F' < F$ **then**
            return false

```
        else
            C_e = C_e − 1
        end if
    end for
    return true
end function
```

3.2 Prove the correctness

假设图G还存在一个最小割E_{cut}.，那么E_{cut}.中至少有一条边e=(u,v)不在ford-folkson算法求出的最小割E里。如果E是唯一的最小割，那每次增加C_e，最大流值F都会增加。但是如果还有最小割E_{cut}.，增加e=(u,v)的C_e，则可以选择不在E 中的边，最大流值F 不会增加。因此如果F'都大于F则最小割E是唯一的。

3.3 time complexity

ford-folkson方法时间复杂度为O(mC)，E_{cut}中的边数为常量K。$T(n) = O(K(mC + c1) + c2) = O(mC)$

方法二：

3.4 Algorithm Description and pseudo-code

自然语言：先对图G求一次最大流，然后在残留网络中分别从源点和终点开始做一次dfs，找出最小割[S,T]，如果[S,T]不包含所有点，那么最小割不唯一。

```
function UniqueCut(Graph, n)
    (S,T) = ford-folkson(Graph)
```

```
    for i = 1 → n do
        vis[i] = 0
    end for
    n1= dfs(S)
    n2= dfs(T)
    if n == n1 + n2 then
        return true
    else
        return false
    end if
end function
```

3.5 Prove the correctness

假设点i不被[S,T]包含，那么残留网络中s不能到达i , i不能到达t,即进入i的边和从i 出去的边都满流，假设某条进入i 的边x 满流，这些流量从若干条边y 流出i ，那么，如果选x 为割边，或者选所有对应的y 为割边，不会影响最大流，即最小割容量不变，最小割也就不唯一。

3.6 time complexity

ford-folkson方法时间复杂度为O(mC)，2次dfs操作，dfs的时间复杂度为O(n+m)，$T(n) = O(mC)$

4 Programming(8)

4.1 result analysis

本题采用Dinic算法实现ford-fulkerson求最大流方法。中间步骤如下：
(1) 对题目构建图模型。本题设定起点s和终点t，每个工作为一个点，将s和每个工作点连接一条边，流值设定为1。每个电脑为一个点，将每个工作和其可以选择的电脑连接一条边，流值设定为1。将每个电脑和t连接一条边，

流值设定为二分查找传入的参数值mid。
(2) 用类似bfs方法对原图构建层次网络。
(3) 得到分层网络后用dfs寻找增广路，不断增广至没有路径，即可得到该图的最大流。
(4) 判断最大流是否满足题目设定，如果最大流值等于job数，则当前构造的图可以完成所有的工作。

problem1的运行结果如下：

5 Programming(9)

5.1 result analysis

本题采用push-relabel算法求最大流，以计算符合条件的矩阵，然后对矩阵是否正确进行检验。输出结果保存在文件out.txt中，包含原图的邻接矩阵，图的最大流值，计算出的合法矩阵结果，正确性检验结果。
中间步骤入下：
(1) 对题目构建图模型。本题设定起点s和终点t，将s与R个代表行和的点连接一条边，流值设为该行的和值，将C个代表列和的点与t连接，流值设定为该列的和值，将行点和列点一一连接，代表R*C个矩阵元素，流值初值设为1。用邻接矩阵表示图。
(2) 用push-relabel算法对构建的图求最大流。设置预流，将源点的全部货物

余量流出。
(3) 搜索除源点和终点以外的节点，找到货物余量大于0的节点做push。没有可以push的顶点则relabel节点u，将其h提高。直到除源点和终点外没有顶点的货物余量大于0。
(4) 输出中间R*C条边的反向流值结果，即结果矩阵。检验结果矩阵正确性：分别计算其各行与各列值的和，对比题目所给数据，输出判断结果（经过检验结果矩阵全部正确）。