专硕《算法设计与分析》复习提要

陈卫东（chenwd@scnu. edu. cn）

2024-12

考试题型：

一、算法分析题（20分）

二、算法设计题（20分）

三、NP完全理论的基本概念（10分）

四、归约技术（20分）

五、NP难度问题的算法设计（20分）

六、并行算法设计（10分）

一、算法分析基础知识

1．解递归方程：$T(n)=3T(n/2)++n\ln n$，　$n>1$；$T(1)=2$。

解答：　由于 $n\ln n=O(n^{\log 3-0.01})$，根据主定理，$T(n)=\Theta(n^{\log 3})$。

注：本题也可使用其它方法（比如数学归纳法）求解。

2．解递归方程：$T(n)=T(n/2)+T(n/4)+cn$，其中 $n>1$, c 为常数；$T(1)=1$。

解答：　使用递归树，$T(n)=cn+\frac{3}{4}cn+(\frac{3}{4})^2cn+...=[1+\frac{3}{4}+(\frac{3}{4})^2+...]\,cn=\Theta(n)$。

注：本题也可使用其他方法（比如数学归纳法）求解。

3．用积分近似求和的方法求函数 $f(n)=\sum_{j=1}^{n}\ln(n/j)$ 的渐近表示（用符号Θ表示）。

解答：考虑增函数 $\ln x$ 在区间$[1,n]$上的定积分及几何意义。

$$\sum_{j=1}^{n}\ln(n/j)=n\ln n-\sum_{j=2}^{n}\ln j\le n\ln n-\int_1^n\ln x\mathrm{d}x$$
$$=n\ln n-(x\ln x|_1^n-\int_1^n x(1/x)\mathrm{d}x)=n-1$$
$$\sum_{j=1}^{n}\ln(n/j)=(n-1)\ln n-\sum_{j=2}^{n-1}\ln j\ge(n-1)\ln n-\int_2^n\ln x\mathrm{d}x$$
$$=(n-1)\ln n-(x\ln x|_2^n-\int_2^n x(1/x)\mathrm{d}x)$$
$$=n-\ln n+2\ln 2-2\le 2n$$

因此，　$n-1\le\sum_{j=1}^{n}\ln(n/j)\le 2n$。即，$\sum_{j=1}^{n}\ln(n/j)=\Theta(n)$。

4．设原问题规模是 n，下述三算法中选择一最坏情况下时间复杂度最低的算法，并说明理由。

算法 A：将原问题划分规模减半的 5 个子问题，递归求解每个子问题，然后再线性时间内将子问题的解合并得到原问题的解。

算法 B：先递归求解 2 个规模为 $n-1$ 的子问题，然后在常数时间内将子问题的解合并得到原问题的解。

算法 C：将原问题划分规模为 $n/3$ 的 9 个子问题，递归求解每个子问题，然后在$\Theta(n^2)$时间内将子问题的解合并得到原问题的解。

解答：算法 ***A*** 时间复杂度 $T(n)$满足 $T(n)=5\mathrm{T}(n/2)+\Theta(n)$，由主定理，$T(n)=\Theta(n^{\log 5})$。

算法 ***B*** 时间复杂度 $F(n)$满足 $F(n)=2F(n-1)+\Theta(1)$，根据主定理，$F(n)=\Theta(2^n)$。

算法 ***C*** 时间复杂度 $G(n)$满足 $G(n)=9G(n/3)+\Theta(n^2)$，根据主定理，$G(n)=\Theta(n^2\log n)$。

因此，算法 ***C*** 的复杂度最低。

5．分析下列冒泡排序的时间复杂度。

算法　REC**BUBBLESORT**

输入：　n 个元素的数组 $A[1\ldots n]$。

输出： 按照非降序排列的数组 $A[1\ldots n]$。

```
1.  sorted←false
2.  sort(n)

过程 sort(i)                    //对 A[1…i]排序
1.  if i ≥2 and not sorted then
3.      for j←1 to i–1
4.        if A[j]>A[j+1] then
5.            交换 A[j]与 A[j+1]
6.            sorted ←false
7.        end if
8.      end for
9.      sort(i–1)          //递归对 A[1…i–1]排序
10. end if
```

解答： 设上述递归算法中元素比较的最多次数 T(n)满足的递归方程为：

T(n)= T(n–1)+ n–1， n≥2； T(1)=0。

使用展开法解得 T(n)=1+2+….+ n–1=Θ(n^2)。

【主定理】设a≥1, b>1为常数，$f(n)$为函数，$T(n)$为非负整数，且

$$T(n)=aT(n/b)+f(n),$$

则有以下结果：

1. 若$f(n)=O(n^{\log_b a-\varepsilon}),\varepsilon>0$，那么$T(n)=\Theta(n^{\log_b a})$.
2. 若$f(n)=\Theta(n^{\log_b a})$，那么$T(n)=\Theta(n^{\log_b a}\log n)$.
3. 若$f(n)=\Omega(n^{\log_b a+\varepsilon}),\varepsilon>0$，且对某个常数$c$<1和所有充分大的$n$有 $af(n/b)\le cf(n)$，那么

$$T(n)=\Theta(f(n)).$$

二、算法设计基本方法

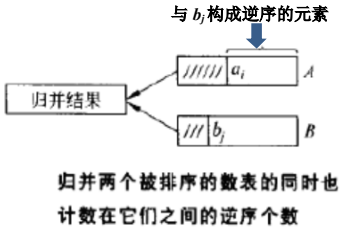
1.（1）可利用归并排序算法的思想设计算法计算 1,2,…,n 的任何排列 $i_1,i_2,\ldots,i_n$ 的逆序个数。请描述算法设计思路，并分析算法复杂度。

（2）画图表示（1）中所设计算法求解 8 个数的排列 3,5,2,7,8,1,4,6 的逆序个数的主要过程。

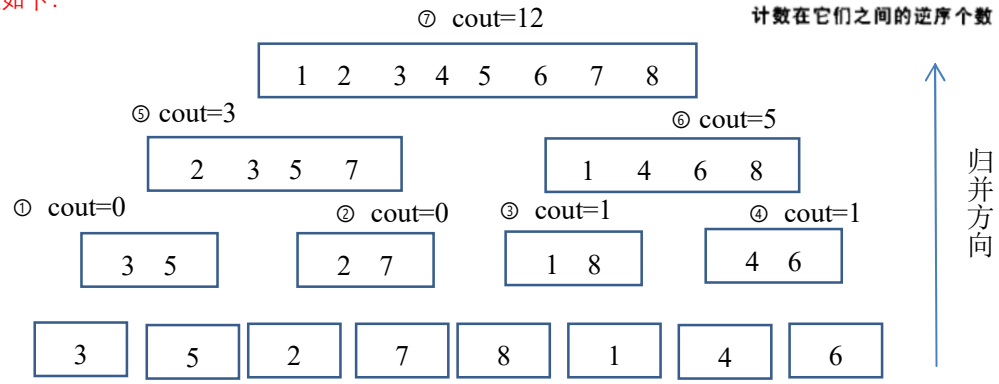
解答：

（1）需要对归并排序算法中的归并过程进行修改，使之在归并两个有序数组的同时也统计出它们之间的逆序个数。修改思路如下：

设置计数器 cout 用于统计排序过程中记录的逆序个数，其初始值为 0。当对两个升序的子数组 $A[a_1,a_2,..,a_p]$和 $B[b_1,b_2,..,b_q]$进行归并时，依次将两个数组的当前元素 a_i 与 b_j 进行比较，值小的取走并依次放入辅助数组 C 中。此时，如果 $a_i>b_j$，则说明 b_j 与 $a_i,..,a_p$ 均构成逆序，因此 cout 值增加 $p–i+1$。示意图如右。



（2）修改后的归并排序算法求解排列 3,5,2,7,8,1,4,6 的逆序个数的主要过程如下：



2. 有 n 个文件需要存储在磁盘上，第 i 个文件需要 s_i 个字节的存储空间，i=1,2,…,n。磁盘总容量为 C 个字节大小，且 $\sum_{i=1}^{n} s_i > C$。

(1) 若要求存入的文件个数达到最多，那么应该选用哪种算法设计技术？简述算法设计思想，并分析算法最坏情况下的时间复杂度。

(2) 若要求磁盘剩余空间达到最小，那么应该选用哪种算法设计技术（要求算法最坏情况时间复杂度不能是关于 n 的指数函数）？简述算法设计思想，并分析算法最坏情况时间复杂度。

解答: (1) 应该选用贪心算法设计技术。对所有文件按照存储空间大小排序，不妨仍然设为 $s_1 \le s_2 \le \ldots \le s_n$。贪心算法的贪心准则：按照文件所占空间由小到大的次序依次将文件存入磁盘，直到遇到某个文件存不进磁盘为止。

贪心算法最坏情况下运行时间包括：文件排序时间 $O(n\log n)$，之后按照贪心准则依次放入文件的时间 $O(n)$，从而总时间为 $O(n\log n)+O(n)=O(n\log n)$。

（2）应该选用动态规划算法设计技术。设函数 $L[i,j]$表示从 1,2,…,i 号文件中选择文件存入容量为 j 的磁盘的所占最大存储空间，其中 $0 \le i \le n$，$0 \le j \le C$。则 $L[i,j]$的递推计算的公式如下：

当 $0 \le i \le n$，$0 \le j \le C$ 时，$L[i,0]=0$, $L[0,j]=0$;

当 $j<s_i$ 时, $L[i,j]=L[i-1,j]$;

当 $j \ge s_i$ 时, $L[i,j]=\max\{L[i-1,j], L[i-1,j-s_i]+s_i\}$;

根据上述递推计算公式可知，算法最坏时间复杂度为 $O(nC)$。

3.【最大间隔聚类问题】给定非负带权无向图 $G=\langle V,E\rangle$及正整数 $k(1 \le k \le |V|)$，试将 G 划分为 k 个非空子图 $G_1,G_2,\ldots,G_k$ 使得任意两子图的最小间距最大。这里两子图的间距是指跨于这两子图之间的最小权的边的权值。

——贪心算法(可用最小生成树问题的 Kruskal 算法求解，在添加后 $k-1$ 条边之前停止即可)。

4．无向图 $G=\langle V,E\rangle$是二部图是指该图的点集 V 能划分成非空的两部分 V_1 和 V_2 使得图 G 中任何边的两个端点分别属于 V_1 和 V_2。考虑用图的**深度优先遍历**方法设计一个算法判别无向图 $G=\langle V,E\rangle$是否是二部图。要求：

（1）用文字简要描述算法的基本思路（不用写代码）。

（2）给出算法正确性的证明。

（3）如果要求利用**宽度优先**遍历图的方法设计算法判别图 $G=\langle V,E\rangle$是否是二部图呢?

——**（1）**基于深度优先遍历图的算法思路如下： 通过从某点开始对图进行深度优先遍历来依次给图中相邻节点分别着黑、白两种颜色（节点的“访问”即为节点的“着色”）。在此过程中如果不存在有两个端点同色的**回边**，则断定该图不是二部图，否则该图是二部图。

（2）上述算法使得图中所有边被分成“树边”和“回边”，且所有树边的两端点是不同色的。

如果图 G 是二部图$\langle V_1 \cup V_2,E\rangle$，则其所有边都跨于 V_1 和 V_2 之间，且算法中给 V_1 中所有点是同一种颜色，给 V_2 中所有点是另一种颜色。因此，不存在有两个端点同色的回边。

反之，若不存在有两个端点同色的回边，下证图 G 是二部图。将 G 中所有节点按颜色归类：所有黑色点的集合记作 V_1，所有白色点的集合记作 V_2。既然 G 中树边都跨于 V_1 和 V_2 之间且没有两端点同色的回边，说明所有边都跨于 V_1 和 V_2 间，即图 G 是二部图。

（3）采用宽度优先遍历图 G，得到宽度优先生成树。如果宽度优先生成树的同层节点没有边出现，则图 G 为二部图，否则图 G 不是二部图。在图 G 的宽度优先生成树中，G 中任何边必跨于相邻层节点之间或同层节点之间，前者是树边，后者是横跨边。若宽度优先生成树的同层节点没有边，则图 G 为二部图。反之，若图 G 是二部图，则必有其宽度优先生成树同层节点之间无边。算法正确性证毕。

5. 使用动态规划法求解下列 0-1 背包问题实例：物品个数 n=4，重量 W={2,3,4,5}，利润 P={3,4,5,7}，背包承重量 C=9。

（1）给出递推计算的公式。

（2）用表格或图展示出主要计算过程，并指出问题的解。

解答：（1）递推计算公式：

令 V[i][j]表示在物品 1,2,…,i 中挑选物品装入容量是 j 的背包中的最大价值。则对于 1≤i≤n, 1≤j≤C，有如下递推计算公式：

V[i][0]=0, V[0][j]=0,

V[i][j]=V[i-1][j] （j<W[i]），

V[i][j]=max{V[i-1][j],V[i-1][j-W[i]]+P[i]} （j≥W[i]）。

（2）计算过程：

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	3	3	3	3	3	3	3
2	0	0	3	4	4	7	7	7	7	7
3	0	0	3	4	4	7	8	9	9	12
4	0	0	3	4	5	7	8	10	11	12

由此得最优装包价值为 12。最优的装包方法为：(0, 0, 1, 1）或 (1,1,1,0)。

6．设有 n 项作业的集合 $J=\{1,2,\ldots,n\}$，每项作业 i 的加工时间 $t_i\in Z^+$,$t_1\le t_2\le\ldots\le t_n$，效益值为 v_i，任务的结束时间为 $d\in Z^+$。一个可行调度是对 J 中作业子集 A 的一个安排，即对 A 中每个作业 i 给定一个开始时间 f_i，满足对任意 $i,j\in A$ 且 $i\neq j$ 有

$f_i+t_i\le f_j$ 或者 $f_j+t_j\le f_i$，且$\sum_{k\in A}t_{kj}\le d$

设机器从 0 时刻开动，只要有作业就不闲置，求具有最大总效益的调度。请设计动态规划算法求解该问题。

（1）说明算法的设计思想（不用写代码），并写出递推计算公式。

（2）分析算法最坏情况下的时间复杂度。

——与 0-1 背包问题的求解类似

7．针对 0-1 背包问题，可以使用动态规划、贪心法、回溯法等算法设计技术来设计其求解算法。请简要说明用这三种技术设计算法的思路以及所得算法的性能特征（包括最坏时间复杂度、所求解是否为最优解等）。

——略。分治法、动态规划法、回溯法都是大问题化为小问题求解，要注意它们的联系和区别。

三、NP 完全性理论基本概念

1．如何理解求解问题的具体算法的复杂度、问题的复杂度?

2．说明语言类 **P**、**NP**、**NP**-complete、**NP**-hard 的含义。假定 **P≠NP**，画图表示其包含关系。

3．什么是问题间的归约? （多项式时间）归约在建立问题的复杂度方面有何作用?

4．证明一个问题是 **NP** 难度有哪些常用方法?

5．如何对付 **NP** 难度问题?

6．指出下面关于 **P≠NP** 错误证明的错处，并说明你认为它是错误的理由。

证明：考虑 **SAT** 的一个算法：“在输入 ϕ 上，尝试变量的所有可能的赋值，若有满足 ϕ 的就接受”。该算法显然需要指数时间。所以 **SAT** 有指数时间复杂度，因此 **SAT** 不属于 **P**。因为 **SAT** 属于 **NP**，所以，**P** 不等于 **NP**。

——错在混淆 SAT 问题的时间复杂度和其一个具体算法时间复杂度。

7．n 皇后问题可以归约为在一个 n^2 点无向图中找一个 n 点独立集的问题（如何归约？）。

既然图的独立集问题是 NP 难度的，能否由该归约得出 *n* 皇后问题是 NP 难度的？请简要说明原因。

——不能。归约方向不对。其实皇后问题属于 P 类，因为存在有多项式时间求解算法（可递归构造出解）。

四、归约技术

1. 工厂因为开展新业务打算明年年初新购置某台设备，并要做好后续 5 年该设备的更新计划,决定每年是重新购置该设备还是继续维护现有设备。已经预测今后 5 年该设备的购置费和维护费分别如下表 1 和表 2 所示(购置价格和维护费用的单位：万元)。需要设置一个该设备的 5 年更新方案使得在此期间该设备的购置费和维护费的总和最小。

请将该问题归约为图的最短路径问题求解。给出思路，并根据表 1 和表 2 中数据给出对应的的图最短路径问题的实例。

表 1　每年的设备购置费用(单位：万元)

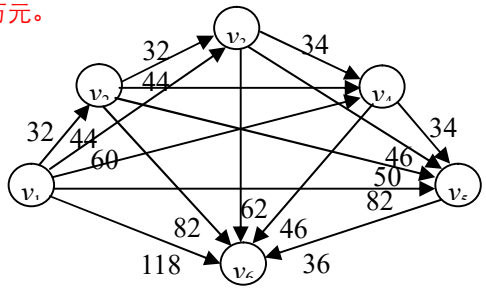
第 1 年	第 2 年	第 3 年	第 4 年	第 5 年
22	**22**	**24**	**24**	**26**

表 2　设备维护费用(单位：万元)

0-1 年	1-2 年	2-3 年	3-4 年	4-5 年
10	**12**	**16**	**22**	**36**

解答： 令 v_i 表示第 i 年初设备是新购置的状态(i=1,2,3,4,5,6)。对于任意 $i<j$，v_i 到 v_j 连有向边，表示第 i 年初新购置设备使用一直使用到第 j 年才又重新购置,其权重表示第 i, i+1, … , j-1 年的设备购置及维护总费用。每条有向边的权重根据上述表格数据计算出来。例如，有向边<v_1,v_4>的权重表示从第 1 年购置新设备使用到 3 年底的总费用，即 w_{14}=22+10+12+16=60，有向边<v_2,v_6>的权重表示从第 2 年购置新设备使用到 5 年底的总费用，即 w_{26}= 22+10+12+16+ 22=82。由此得到一个下图所示的有向赋权图。

原问题中寻找总费用最少的设备更新方案问题由此转为求从 v_1 到 v_6 的最短路径问题。容易看出，图中有两条最短路径：$v_1\to v_3\to v_6$, $v_1\to v_4\to v_6$ 分别表示第 1 年和第 3 年初需购置新设备的方案和第 1 年和第 4 年初均购置新设备的方案,总费用均为 106 万元。



2. 设 P 是一台 Internet 上的 Web 服务器，T={1,2,…,n}是 n 个下载请求的集合,$\forall i\in T$, $a_i\in Z^+$表示下载请求 i 所申请的带宽．已知服务器的最大带宽是正整数 K．我们的目标是使得带宽的最大限度的利用,即确定 T 的一个子集 S 使得$\sum_{i\in S}a_i\le K$,且 $K-\sum_{i\in S}a_i$ 达到最小。设计一个算法求解服务器下载问题，用文字说明算法的主要设计思想和步骤，并给出最坏情况下的时间复杂度。

解答： 实际上这是一个 0-1 背包问题实例：第 i 个物品价值和大小均为第 i 个下载请求的带宽 a_i（$1\le i\le n$），背包容量为最大带宽 K。用动态规划算法求解。设函数 $V[i,j]$表示从 1,2,…,i 号物品中选择物品装入容量为 j 的背包的最大价值，并用标记函数 $I[i,j]$来追踪最优的装包方法，其中 $1\le i\le n$。则

$V[i,j]$的递推计算公式如下：

当 $0\le i\le n$，$0\le j\le B$ 时，$V[i,0]=0$, $V[0,j]=0$;

当 $j<w_i$ 时，$V[i,j]=V[i-1,j]$;

当 $j\ge w_i$ 时，$V[i,j]=\max\{V[i-1,j], V[i-1,j-w_i]+p_i\}$;

标记函数 $I[i,j]$计算公式如下：

(i) 如果 $V[i,j]=V[i-1,j]$，则 $I[i,j]=0$（表示最优方案中第 i 号物品不放）;

(ii) 如果 $V[i,j]>V[i-1,j]$，则 $I[i,j]=1$（表示最优方案中第 i 号物品放入背包）。

根据上述递推计算公式可知，算法的时间复杂度为 $O(nK)$。

3．证明: **点覆盖问题≤Roman-Subset 问题**

【点覆盖问题】给定无向图 $G=\langle V,E\rangle$，求最小规模点子集 $S\subseteq V$ 满足 S 能覆盖 E 中所有边。

【Roman-Subset 问题】给定一个有向图 $G=\langle V,E\rangle$，求最小规模点子集 $S\subseteq V$ 满足 G 中任何回路上都至少有一点在 S 中。

——归约要点：给定**点覆盖问题**的一个实例，即一个无向图$G=\langle V,E\rangle$。构造**Roman-Subset问题**的实例如下：将G中每条无向边$(u,v)\in E$变成两条有向边$<u,v>$和$<v,u>$，所得有向图记作$G^*=\langle V,E^*\rangle$（G中每条边变为长度为2的回路）。显然，G的一个k点覆盖也是G^*的一个k点Roman Subset。反之，对于G^*的一个k点Roman Subset一定与每条长度为2的回路（原图G中的每条边）有共同点，即它必是G的一个k点覆盖。

4．试根据最大独立集问题的计算复杂性，证明兴趣互异顾客群问题是NP难度的。根据给定的最大独立集问题的实例图，画出归约示意图。

某商店为了分析它的顾客的行为，要经常维护一个 2 维数组 A，其中行对应它的顾客，列对应它出售的商品. 元素 $A[i,j]$是顾客 i 购买商品 j 的数量.

下面是一个很小的这样的数组 A

	液体清洁剂	啤酒	尿布	猫用干草
Raj	0	6	0	3
Alanis	2	3	0	0
Chelsea	0	0	0	7

商店可能想用这些数据做下面的事情. 如果顾客子集 S 中的任意两位顾客从来没有买过相同的商品(即，对每一种商品，S 中至多有一位顾客买过)，则称 S 是兴趣互异的. 兴趣互异的顾客集合可能是有用的，例如，它可能集中了市场调查的研究对象.

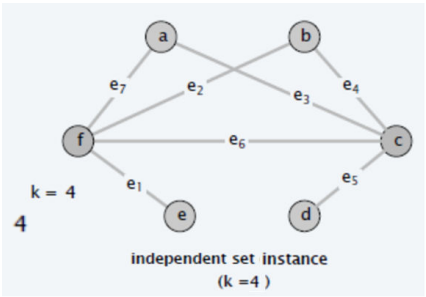
兴趣互异的子集问题定义如下：给定如上定义的一个 $m\times n$ 数组 A 和一个数 $k\leqslant m$，有至少有 k 位顾客的兴趣互异的子集吗?

——归约要点：给定独立问题的一个实例，即给定无向图$G=<V,E>$和正整数k，问G中是否存在至少有k个点的独立集。据此，我们构造一个兴趣互异顾客群问题实例如下：图G中每个点对应一个顾客，G中每条边对应一种产品，如果边e与点v关联，就构造二维数组A中的一行来表明顾客v购买了产品e。问：是否二维数组A中有至少k位顾客的兴趣是互异的。

显然，图G中存在点数至少为k的独立集当且仅当二维矩阵A中存在至少k顾客的兴趣是互异的。

上述归约过程能在多项式时间内完成。因此，独立集问题多项式归约为兴趣互异顾客群问题。由此表明，兴趣互异顾客群问题的难度不低于图的独立集问题的难度。因独立集问题是NP难度问题，所以兴趣互异顾客群问题也是NP难度问题。

从独立集问题到兴趣互异子集问题的归约示意图：



	e_1	e_2	e_3	e_4	e_5	e_6	e_7
a	-	-	1	-	-	-	1
b	-	1	-	1	-	-	-
c	-	-	1	1	1	1	-
d	-	-	-	-	1	-	-
e	1	-	-	-	-	-	-
f	1	1	-	-	-	1	1

五、NP-hard 问题的算法设计

1.【并行调度问题】设有 n 项任务由 p 个可并行操作的的机器完成，完成任务 i 所需要的时间是 $t_i>0$，求一个使得完成时间（即从时刻 0 开始计时到最后一台机器停止的时间）达到最短的分配方案。

【负载平衡问题】设有n个独立的作业$J_1, J_2,\ldots, J_n$需要处理，其中作业J_i需要t_i个机时（i=1,2,…,n）。现只有p台完全相同的机器$\boldsymbol{M}_1, \boldsymbol{M}_2,\ldots, \boldsymbol{M}_p$可以同时使用，任何一项作业可在任一台机器上加工。现在需要对这些作业进行安排使得所有机器的负载尽可能的均衡。

（1）试证明负载均衡问题是**NP-hard**问题。

——要点：将Partition归约到负载均衡问题，或SubsetSum归约到负载均衡问题即可。

（2）下面是求解负载平衡问题的一个简单的贪心算法：

算法 Greedy-Balance

1. 初始化： 所有的作业没有被安排

   对第i台机器M_i，令T_i=0，$A(i)$=∅

   //T_i表示机器i上所有作业的处理总时间，$A(i)$表示分配给机器i的作业集合

2. 按处理时间t_i的非增次序给作业排序，不妨设 $t_1 \geq t_2 \geq \ldots \geq t_n$

3. **FOR** j =1 ,…, n **DO**

4. 设机器M_i达到最小值$\mathbf{min}_k\, T_k$

5. 把作业j分配给机器M_i

6. $A(i) \leftarrow A(i) \cup \{j\}$

7. $T_i \leftarrow T_i + t_j$

7. **END-FOR**

试证明该算法是3/2-近似算法。

——**证明:** (i) 如果负载最大的机器上只有一个作业，则显然贪心解值A(I)=O(I)，即A(I)/O(I)=1<3/2；

(ii) 如果负载最大的机器上至少有2个作业，根据贪心算法思路必有最后加入作业的处理时间$t_j \leq t_{p+1}$ ($j \geq p$+1). 注意到O(I) ≥2 $t_{p+1} \geq 2t_j$. 因此，考虑到该机器上的负载A(I)分为两部分：

最后一项作业之前的负载 A(I)- $t_j \leq (\Sigma_{1\leq i\leq n}\, t_i - t_j)/p$，即有 $A(I) \leq \Sigma_{1\leq i\leq n}\, t_i/p + (1-1/p)\, t_j$；

最后一项作业的处理时间 $t_j \leq$ O(I)/2. (注意到$\Sigma_{1\leq i\leq n}\, t_i/p \leq$ O(I))

因此，有 A(I) ≤ O(I)+(1−1/p) O(I)/2= O(I)(3/2− 1/(2p)) < O(I)3/2. 即有 A(I)/O(I)<3/2.

（3）请通过设计该问题的一个多项式近似方案来证明它属于 **PTAS**。

——参见近似算法授课讲义。

2.【两机并行调度问题】设有n项任务，加工时间分别表示为正整数$t_1, t_2,\ldots, t_n$。现有两台同样的机器，从0时刻开始安排对这些任务的加工，规定只要有待加工的任务，任何机器都不得闲置。如果直到时刻T所有任务都完成了，总的加工时间就等于T。要求找出使得总的加工时间T达到最小的调度方案。

两机并行调度问题其实可建模为 0-1 背包问题，从而可利用动态规划算法求解。请给出建模思路。给出下列实例的计算过程：n=5, t_1=1，t_2=5, t_3=2, t_4=10, t_5=3。

解答: 令 $B=\lceil (t_1+t_2+\ldots+ t_n)/2 \rceil$，则一定存在一个最优调度方案使得第一台机器上安排的任务加工时间总和 $s \leq B$，且 s 到达最大；反之亦然。实际上这是一个 0-1 背包问题实例：第 i 个物品价值和大小均为 t_i（$1 \leq i \leq n$），背包容量为 B。因此，用（1）中动态规划算法求得该 0-1 背包问题实例的最优解，将最优解中对应任务安排在第一台机器上，并将剩下所有任务安排在第二台机器上，即得到两机调度问题的一个最优调度方案。）

针对给定实例（n=5, t_1=1, t_2=5, t_3=2, t_4=10, t_5=3），则 $B=\lfloor (t_1+t_2+\ldots+ t_n)/2 \rfloor$=10，二维数组 V[0…5, 0…10] 的计算过程如下：

	j=0	1	2	3	4	5	6	7	8	9	10
i= 0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1	1	1	1	1
2	0	1	1	1	1	5	6	6	6	6	6
3	0	1	2	3	3	5	6	7	8	8	8
4	0	1	2	3	3	5	6	7	8	8	10
5	0	1	2	3	4	5	6	7	8	9	10

由此得到问题的最优解为：(0,1,1,0,1)和(0,0,0,1,0)，最优值为 10。

因此，最优调度方案为第一台机器上安排任务 2,3,5 且第二台机器上安排任务 1 和 4，或第一台机器上安排任务 4 且第二台机器上安排任务 1,2,3,5。这两种最优方案的加工时间均为 11。

3.【广告策略问题】给定图 $G=\langle V, E\rangle$,路径集 $P_1, P_2,\ldots, P_t$，试找出图中一个最少元素的点集 S,满足在 S 中所有点上放置广告能使得每一条路径 $P_i(1\le i\le t)$上至少有一个结点放置了广告。试给出该问题的一个近似算法。

——将广告策略问题归约到集合覆盖问题，然后用集合覆盖问题的贪心近似算法求解广告策略问题即可。

4.【影响最大化问题】给定有向图 $G=\langle V, E\rangle$，其中每个结点 $v\in V$ 有一个阈值 $w(v)\in[0,1]$,以及参数 $k(1\le k\le|V|)$，试在图中找出 k 个点的子集 S 使得 S 作为信息源能使信息在图中得到最大范围的传播。试给出求解该问题的一个启发式算法。

——贪心启发式算法、迭代改进型启发式算法（局部搜索、禁忌搜索、模拟退火、…）

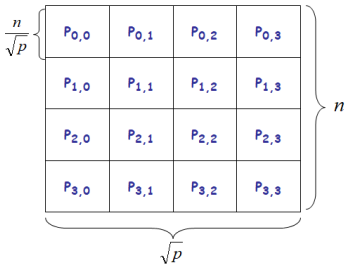
5. 【卫兵布置问题】一个博物馆由排成 $n\times m$ 个矩形阵列的陈列室组成，需要在陈列室中设立哨位，每个哨位上哨兵除了可以监视自己所在陈列室外，还可以监视自己所在陈列室的上、下、左、右四个陈列室。试给出一个最佳的哨位安排方法，使得所有陈列室都在监视之下，但使用的哨兵最少。试证明该问题是多项式时间可解答，并给出一个多项式时间复杂度的算法。

——证明网格图是二部图，由此表明其最小点覆盖大小等于最大匹配数。

六、并行算法设计

矩阵乘法——Cannon乘法

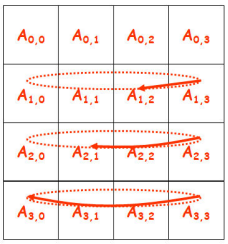
- 分块：A、B和C分成 $p=\sqrt{p}\times\sqrt{p}$ 的方块阵$A_{i,j}$、$B_{i,j}$和$C_{i,j}$, 大小 $\frac{n}{\sqrt{p}}\times\frac{n}{\sqrt{p}}$ 均为p个处理器编号$(P_{0,0},\ldots,P_{0,\sqrt{p}-1},\ldots,P_{\sqrt{p}-1,\sqrt{p}-1})$ $P_{i,j}$存放$A_{i,j}$、$B_{i,j}$和$C_{i,j}(n>>p)$.



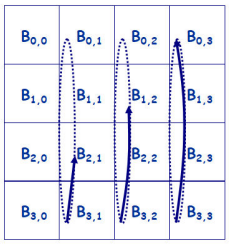
- 算法原理(非形式描述)

① 所有块$A_{i,j}(0\le i,j\le\sqrt{p}-1)$向左循环移动i步(按行移位);
所有块$B_{i,j}(0\le i,j\le\sqrt{p}-1)$向上循环移动j步(按列移位);
② 所有处理器$P_{i,j}$做执行$A_{i,j}$和$B_{i,j}$的乘-加运算;
A的每个块向左循环移动一步;
B的每个块向上循环移动一步;
③ 转②执行 $\sqrt{p}$ 次;

矩阵A的初始对准



矩阵B的初始对准



- 算法时间 $T_p(n)=O(n^3/p)$

- 示例 (这里设处理器个数 p=9，且 $p|n$，处理器排列为 3×3 的方阵$(P_{ij})_{3\times3}$)

$$\begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix}$$

①所有块 $A_{i,j}(0\le i,j\le \sqrt{p}-1)$向左循环移动 i 步(按行移位)；
所有块 $B_{i,j}(0\le i,j\le \sqrt{p}-1)$向上循环移动 j 步(按列移位)；

$$\begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{11} & A_{12} & A_{10} \\ A_{22} & A_{20} & A_{21} \end{bmatrix} \quad \begin{bmatrix} B_{00} & B_{11} & B_{22} \\ B_{10} & B_{21} & B_{02} \\ B_{20} & B_{01} & B_{12} \end{bmatrix}$$

P_{00}： $A_{00}B_{00}$， **P_{01}：** $A_{01}B_{11}$， **P_{02}：** $A_{02}B_{22}$
P_{10}： $A_{11}B_{10}$， **P_{11}：** $A_{12}B_{21}$， **P_{12}：** $A_{10}B_{02}$
P_{20}： $A_{22}B_{20}$， **P_{21}：** $A_{20}B_{01}$， **P_{22}：** $A_{21}B_{12}$

②第一次：A 的每个块向左循环移动一步； B 的每个块向上循环移动一步；

$$\begin{bmatrix} A_{01} & A_{02} & A_{00} \\ A_{12} & A_{10} & A_{11} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \quad \begin{bmatrix} B_{10} & B_{21} & B_{02} \\ B_{20} & B_{01} & B_{12} \\ B_{00} & B_{11} & B_{22} \end{bmatrix}$$

P_{00}： $A_{01}B_{10}$， **P_{01}：** $A_{02}B_{21}$， **P_{02}：** $A_{00}B_{02}$
P_{10}： $A_{12}B_{20}$， **P_{11}：** $A_{10}B_{01}$， **P_{12}：** $A_{11}B_{12}$
P_{20}： $A_{20}B_{00}$， **P_{21}：** $A_{21}B_{11}$， **P_{22}：** $A_{22}B_{22}$

第二次：A 的每个块向左循环移动一步； B 的每个块向上循环移动一步；

$$\begin{bmatrix} A_{02} & A_{00} & A_{01} \\ A_{10} & A_{11} & A_{12} \\ A_{21} & A_{22} & A_{20} \end{bmatrix} \quad \begin{bmatrix} B_{20} & B_{01} & B_{12} \\ B_{00} & B_{11} & B_{22} \\ B_{10} & B_{21} & B_{02} \end{bmatrix}$$

P_{00}： $A_{02}B_{20}$， **P_{01}：** $A_{00}B_{01}$， **P_{02}：** $A_{01}B_{12}$
P_{10}： $A_{10}B_{00}$， **P_{11}：** $A_{11}B_{11}$， **P_{12}：** $A_{12}B_{22}$
P_{20}： $A_{21}B_{10}$， **P_{21}：** $A_{22}B_{21}$， **P_{22}：** $A_{20}B_{02}$

汇总：**P_{00}：** $A_{00}B_{00}+A_{01}B_{10}+A_{02}B_{20}$， **$P_{01}$：** $A_{01}B_{11}+A_{02}B_{21}+A_{00}B_{01}$， **$P_{02}$：** $A_{02}B_{22}+A_{00}B_{02}+A_{01}B_{12}$
P_{10}： $A_{11}B_{10}+A_{12}B_{20}+A_{10}B_{00}$， **$P_{11}$：** $A_{12}B_{21}+A_{10}B_{01}+A_{11}B_{11}$， **$P_{12}$：** $A_{10}B_{02}+A_{11}B_{12}+A_{12}B_{22}$
P_{20}： $A_{22}B_{20}+A_{20}B_{00}+A_{21}B_{10}$， **$P_{21}$：** $A_{20}B_{01}+A_{21}B_{11}+A_{22}B_{21}$， **$P_{22}$：** $A_{21}B_{12}+A_{22}B_{22}+A_{20}B_{02}$