习题参考答案

习题 3.1

1. 编程实现 PlanarColor(V, E)并分析其复杂度。
   略
2. 请给出渐进近似比的直观含义。
   对于一些问题，当问题规模不是很大时，相对近似比很难找到一个界或相对近似比很大，甚至无上界，但当规模较大时，近似比总存在一个上界，这种情况就可以用渐进近似比来刻画。
   总之，渐进近似比描述的是相对近似比的趋势，从数学角度讲就是极限。

习题 3.2

1. 找到一个任务的集合，验证 $(2-1/m)$ 是 LS 算法能够达到的最好的近似比。（提示：考虑一个有 $m(m-1)$ 个长度为 1 的工件，之后又紧跟一个较长的作业的列表）。

   设机器数目为 m，考虑有 $m(m-1)$ 个长度为 1 的任务和一个长度为 m 的任务。则 $A(I)= m-1+m$，OPT(I)= m （在一台机器上放置长度为 m 的任务，其他 m-1 台机器每台放置 m 个长度为 1 的任务），因此，LS 调度可以达到的最好的性能比率为 $(2-1/m)$。

2. 找出到一个任务的集合，使它能够表明 $(4/3-1/3m)$ 是 LPT 算法能够达到的最好的近似比。

   假设有 m 台机器，$2m$+1 条任务，前 $2m$ 个任务长度分别为 $2m-\left\lfloor \frac{i+1}{2} \right\rfloor$，$1\le i\le m$ 最后一条为 m，即作业分别为：
   $$\{2m-1,2m-1,2m-2,2m-2,\cdots,m+1,m+1,m,m,m\}$$
   OPT(I)=3m。
   LPT：

$$第1台处理 2m-\left\lfloor\frac{1+1}{2}\right\rfloor 和 2m-\left\lfloor\frac{2m+1}{2}\right\rfloor=2m-\left\lfloor\frac{m+(m+1)}{2}\right\rfloor=m$$

$$第2台处理 2m-\left\lfloor\frac{2+1}{2}\right\rfloor 和 2m-\left\lfloor\frac{2m-1+1}{2}\right\rfloor=2m-\left\lfloor\frac{m+m}{2}\right\rfloor=m$$

$$第3台处理 2m-\left\lfloor\frac{3+1}{2}\right\rfloor 和 2m-\left\lfloor\frac{2m-2+1}{2}\right\rfloor=2m-\left\lfloor\frac{m+(m-1)}{2}\right\rfloor$$

$$第\mathrm{i}台处理 2m-\left\lfloor\frac{i+1}{2}\right\rfloor 和 2m-\left\lfloor\frac{2m-(i-1)+1}{2}\right\rfloor=2m-\left\lfloor\frac{m+(m-i+2)}{2}\right\rfloor$$

$$第m台处理 2m-\left\lfloor\frac{m+1}{2}\right\rfloor 和 2m-\left\lfloor\frac{2m-(m-1)+1}{2}\right\rfloor=2m-\left\lfloor\frac{m+2}{2}\right\rfloor 以及 \mathrm{m}$$

$$2m-\left\lfloor\frac{i+1}{2}\right\rfloor+2m-\left\lfloor\frac{2m-(i-1)+1}{2}\right\rfloor$$

$$=4m-\left\lfloor\frac{i+1}{2}\right\rfloor-\left\lfloor\frac{2m-i+2}{2}\right\rfloor=4m-\left\lfloor\frac{i+1}{2}\right\rfloor-\left\lfloor\frac{2m+2}{2}\right\rfloor-\left\lfloor-\frac{i}{2}\right\rfloor$$

$$=3m-1-\left(\left\lfloor\frac{i+1}{2}\right\rfloor-\left\lfloor-\frac{i}{2}\right\rfloor\right)=3m-1$$

$$A(I)=4m-1$$

$$\frac{A(I)}{OPT(I)}=\frac{4m-1}{3m}=\frac{4}{3}-\frac{1}{3m}$$

3. 编程实现 LPTc(I)算法。
   略
4. 思考怎么样改进 LPTc(I)算法使得 m 固定的这个假设可以消除。
   参考书中文献。
5. 为最小切割问题设计一个近似算法。

   可以利用贪心算法，先令 V_1 为空集，每次增加一个顶点到 V_1，使得 $|E(V_1, E-V_1)|$ 最小。

   或者利用如下局部搜索算法

   Mincut(G)

   V_1=ø

   repeat

       if exchanging one node between V_1 and V_2=V-V_1 improves the cut **then**

perform the exchange

until a local optimum is reached;

return *f*

上述局部搜索算法是一个 2-近似算法，证明如下

首先我们证明局部最优解至少含有$|E|/2$条边，令 c 表示切割的边数，i 表示集合V_1中的边数，o 表示集合$V-V_1$中的边数，则有$|E|=c+i+o$,即 $i+o=|E|-c$。

对任意顶点v，令 $i(v)$表示连接v以及V_1中顶点的边数，令 $o(v)$表示连接v以及$V-V_1$中顶点的边数。当算法到达局部最优时，对任意的$v\in V_1$，有 $i(v)-o(v) \le 0$ ，同理，对于任意的$v\in |V|-V_1$，有 $o(v)-i(v) \le 0$.

对所有在V_1中的边数求和，有 $2i-c \le 0$

对所有在$V-V_1$中的边数求和，有 $2o-c \le 0$

由上述式子可得 $i+o-c \le 0$，可得$|E|-2c \le 0$，即 $c \ge |E|/2$。而 $\mathrm{OPT(I)} \le |\mathrm{E}|$，故近似比为 2。

习题 3.3

1. 考虑如下的查找一个近似旅行商回路的最近点启发式算法。刚开始选择仅包括任意选择的一个顶点的简单回路。然后，每一步找到一个还不在这个回路中顶点u，且u到回路中的任何顶点的距离最小。假设回路中距离u最近的顶点是v。这时扩展回路来包括u，通过把u插入v的后面。重复执行直到所有顶点都在回路中。证明这个启发式算法返回一个回路，它的权值至多是一个最优化回路的权值的两倍。

   类似书上证明，也可以如下证明：

   证明：假设最优回路为 $H^* = v_1, v_2, \cdots, v_{|V|}$， v_i 和 v_j 是 H^* 中两个相邻的顶点，$W(v_j)$ 为 v_j 插入到回路中增加的权值。

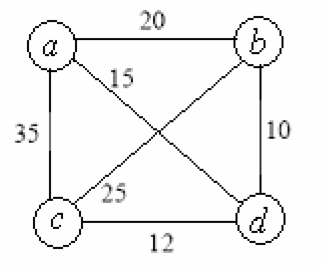
   为了方便证明，增加一个 $v_{|V|+1} = v_1$， $H^* = v_1, v_2, \cdots, v_{|V|}, v_{|V|+1}$， $v_{|V|+1}$ 到任何顶点的距离都为 0。

   根据引理 3.3.1 的结论 $2w(v_i, v_j) \ge \min\{W(v_i), W(v_j)\}$，有：

$$2OPT(I) = 2\sum_{i=1}^{|V|} w(v_i, v_{i+1}) \ge \sum_{i=1}^{|V|} W(v_{i+1}) = \sum_{i=2}^{|V|+1} W(v_i) = \sum_{i=1}^{|V|} W(v_i) = R(I)$$

$$\frac{R(I)}{OPT(I)} \le 2$$

2. 给定一个无向完全图：



请利用最远点插入启发式算法，找出该图的回路。
最远点插入法：插入离已经构建的回路中任一顶点最远的顶点。

假设初始随机选择出发点 a, H=a,a

1）选择一个最远的 c ,H=a,c,a

2) 选择一个最远的 b, H=a,c,b,a

3) 选择一个最远的 d,

$w(a,d)+w(b,d)-w(a,b)=5$

$w(a,d)+w(c,d)-w(a,c)=-8$

$w(b,d)+w(c,d)-w(b,c)=-3$

H=a,d,c,b,a

3. 编程序实现 NearestNeighbor(G)，ShortestLinkedHeuristic(G)以及 NearestInsertion(G)，并通过对随机生成的例子的计算，对这些算法进行比较分析。
略，可到 http://59.77.16.229/下载测试例子。
4. 证明定理 3.3.3。
非常复杂。可以略。

习题 3.4

1. 给出一个有效的贪心算法，在线性时间内找到一棵树的最优顶点覆盖。
思路 1：

1）对树中顶点按层遍历，统计奇层顶点数与偶层顶点数；

2）比较奇层顶点数与偶层顶点数，取小者作为顶点覆盖。

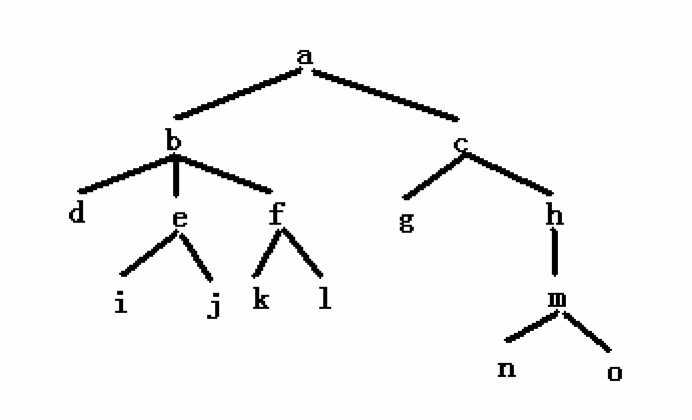
该方法虽可得到线性时间找到一个顶点覆盖，但不是最优的。而且该算法没有体现贪心思想。

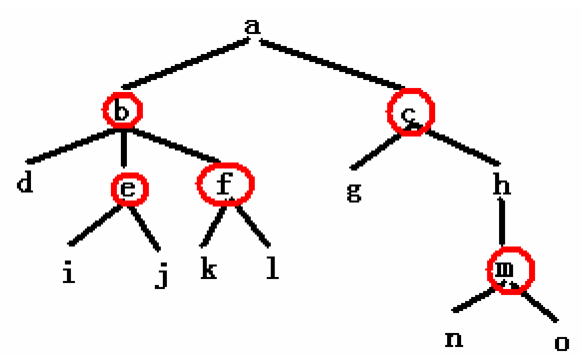
思路 2：在选择顶点覆盖集时，我们更希望选择的是父节点而不是子节点。这就是贪心。

1）对树进行深度优先搜索

2）当从一个孩子节点返回其父节点时，若该孩子节点不在覆盖集 C 中，则将该孩子的父节点加入到 C 中。

举例：





证明：设 C 为思路 2 得到的一个集合，我们只需证明：

1）C 是 T 的顶点覆盖

2）用 C 来构造集合 B，对于任意一个 $u\in C$，都有一条边 $(u,v)\in B$，其中 v 是 u 的孩子，且 v 不在 C 中。这样的 v 总是存在的，因为 u 之所以可以加到 C 中，正是因为至少有一个孩子没有被覆盖到。

3）由 2）知 $|B|=|C|$，且 B 中的所有边没有公共顶点。

首先 B 中任一条边 (u,v)，u 来自 C 中，因此，各条边的 u 是没有公共顶点的，v 是 u 的孩子，树中的任意一个孩子都不可能有两个不同的父亲，因此，v 也是没有公共顶点的。

4）T 的任意一个顶点覆盖，一定要包含 B 中各边的至少一个顶点。

$OPT(I)\geq|B|=|C|=A(I)$，另一方在 $A(I)\geq OPT(I)$，因此，$A(I)=OPT(I)$

2. 令 B 表示 GreedyVertexCover(G)算法第 4 行所选边的集合。证明集合 B 是图 G 的最大匹配。

此题有问题，更正如下（思考为什么要更正？）：令 B 表示 GreedyVertexCover(G)算法第 4 行所选边的集合。证明集合 B 是图 G 的极大匹配。

概念回顾：

1）设 $G=(V,E)$，$E^*\subseteq E$，若 E^* 中任何两条边均不相邻，则称 E^* 是 G 的匹配；

2）若在 E^* 中添加任意一条边，所得集合都不再匹配，则称 E^* 是 G 的极大匹配；

3）边数最多的匹配（极大匹配）称为最大匹配。

证明：B 是极大匹配

1）把算法中每一次删除的边的集合记为 M，则 $B\cup \mathrm{M}=E$

2）任意选择一条不在 B 中的一条边 e，则 e 一定属于 M，而 M 中的任意一条边均可以在 B 中找到与其有公共顶点的边（即邻边），因此 B 是 G 中的极大匹配。

3. 证明贪心顶点覆盖问题的近似比。

1）对于任意一个顶点覆盖实例 $G=(V,E)$，构造集合覆盖问题的实例 $I=(X,F)$：

A．将每个顶点看成一个元素，$X=E$；

B．由每个顶点 v_i 构造出一个子集 s_i，$s_i=\{(u,v)\mid u=v_i\text{或}v=v_i\}$，即 s_i 为与 v_i 相关联的边的集合；

2）把贪心集合覆盖的解 C 作为顶点覆盖问题的解。

由推论 3.4.1 知，贪心算法求集合覆盖问题的的近似比为 $\ln|X|+1$，因此，贪心顶点覆盖问题的近似比也为 $\ln|X|+1$。

4. 设计一个求解带权集合覆盖问题的近似算法。

给定一个包含 n 个元素的集合 X，F 包含 X 的 m 个子集，即 $F=\{S_1,S_2,\cdots,S_m\}$

对 F 中的每个子集 S_i 的权值为 $w(S_i)>0$，问题是找出 F 中的一个子集 C，使得 C 覆盖 X 且总权值最小。

算法思路：模仿背包问题

MinWeightedSetCover(X, F)

$U \leftarrow X$

$C \leftarrow \varnothing;\ i \leftarrow 1$

sort F by non-decreasing ordering of $\dfrac{w(S_i)}{|S_i|}$

while $U \neq \varnothing$ **do**

  if $|S_i \cap C| \neq |S_i|$ **then**

    $U \leftarrow U - S_i$

    $C \leftarrow C \cup \{S_i\}$

    $W \leftarrow W + w(S_i)$

return C

分析上述算法：随着 C 增加，S_i 中未被覆盖的元素越来少，有的 S_i 甚至已全被覆盖过了，但 $\dfrac{w(S_i)}{|S_i|}$ 权值仍比较大，因此不能反映实际需求。

改进:

MinWeightedSetCover(X, F)

$U \leftarrow X$

$C \leftarrow \varnothing$

$M \leftarrow F$

while $U \neq \varnothing$ **do**

select $S \in M$ such that $\min\limits_{1 \le i \le |M|} \{\frac{w(S_i)}{|S_i - S_i \cap C|}\}$

$U \leftarrow U - S$

$C \leftarrow C \cup \{S\}$

$W \leftarrow W + w(S)$

$M = F - S$

return C

5. 证明集合覆盖问题的判定形式是 NP 完全的（提示，可将集合覆盖问题归约到顶点覆盖问题）

证明思路：

1）要证集合覆盖问题属于 NP，即证它是多项式时间可验证的。

要判断 F 的一个子集 C 是否覆盖 X，显然在线性时间内就可完成。

2）利用某一 NP 完全的语言类（顶点覆盖问题、3-SAT），证明其多项式时间可约简到集合覆盖问题。

证明顶点覆盖问题可多项式时间约简到集合覆盖问题

问题转化为：对于任意一个顶点覆盖实例 $G=(V,E)$，如何构造出集合覆盖问题的实例 $I=(X,F)$，使得 I 存在 k 集合覆盖当且仅当 G 存在 k 顶点覆盖。

对于任意一个顶点覆盖实例 $G=(V,E)$，按如下方法构造集合覆盖问题的实例构造集合覆盖问题的实例 $I=(X,F)$：

A．将每个顶点看成一个元素，X=E；

B．由每个顶点 v_i 构造出一个子集 s_i，$s_i=\{(u,v) \mid u=v_i 或 v=v_i\}$，即 s_i 为与 v_i 相关联的边的集合；

1）当 B 是 G 的一个顶点覆盖时，$B=\{v_{b_1},v_{b_2},\cdots,v_{b_k}\},v_{b_i}\in V$，

顶点覆盖集的含义就是 G 中的任意一条边的两端点至少有一个在顶点覆盖集中，

因此 $C=\{s_{b_1},s_{b_2},\cdots,s_{b_k}\}$ 是 $I=(X,F)$ 的集合覆盖。

2）当 I 存在 k 集合覆盖，设 $C=\{s_{b_1},s_{b_2},\cdots,s_{b_k}\}$ 是 I 的一个 k 集合覆盖，

由之前的构造易知 $B=\{v_{b_1},v_{b_2},\cdots,v_{b_k}\}$ 是 G 的一个 k 顶点覆盖。

习题 3.5

1. 证明解 Bin packing 问题的 Next fit strategy 是一个 2－近似算法。

   考虑所谓的 Next Fit 策略：以给定的顺序取物品 $S=(s_1,s_2,\cdots,s_n)$，一个接一个装满箱子。假设 $A(I)$ 个箱子被用来装载所有的 n 个物品，令 b_i 为第 i 个箱子的装载大小，则 $b_i+b_{i+1}>1$，这样，我们得到

$$2\sum_{i=1}^{m} b_i > A(I).$$

   另一方面，OPT(I)箱子足够装下这些物品，因此

$$\sum_{i=1}^{m} b_i \le \mathrm{OPT}(I).$$

   根据上两式可得

$$A(I) < 2\sum_{i=1}^{m} b_i \le 2\mathrm{OPT}(I).$$

2. 对任意给定的实例 I，证明解 Bin packing 问题的 First fit strategy 所需要的箱子数 $A(I)$ 与最优解所需要的箱子数 $\mathrm{OPT}(I)$ 满足 $A(I)\le 2\mathrm{OPT}(I)+1$.

   参考书

3. 分析并行机调度问题与 Bin packing 问题的关系。

   并行机调度问题：用 m 台机器来完成 n 个作业，问至少需要多少时间。

   Bin packing 问题：用给定容量的容器来装 n 个物品，求至少需要多少个容器。

   相同点：两者都要求每个机器（或箱子）分配的任务（或物品）都较平均。

   不同点：相同机器调度问题是机器数目固定，而装箱问题是箱子的最大容量固定。

   1）并行机调度问题→ Bin packing 问题

   把 m 台机器看作 m 个容器，n 个作业看作 n 个物品，完成每个作业所需时间看作是物品的体积，

   问题转化为：用 m 个容器来装 n 个物品，求容器的容量至少要多大。（假设容器的容量是可以变化的）

   2）同理：Bin packing 问题→并行机调度问题

   因此可以看出，两者互为约束和目标。

4. 编程实现 First fit decreasing，Next fit decreasing，Best fit decreasing，Worse fit decreasing，通过对随机产生的例子计算，对它们进行比较分析。

   略，可以从 http://59.77.16.229/下载测试例子。每种策略的思路、执行的性能方面考虑。

5. 为 First Fit 策略设计一个随机近似算法。
   可以先随机对物品排序，然后调用 First Fit 算法

习题 3.6

1. 证明：$V_{\max} \le 2\sum_{i\in S'} v_i$.

2. 证明本节 PTAS 中 $V(G) \ge \sum_{i=k+1}^{m-1} v(a_i) + \Delta \frac{v(a_m)}{w(a_m)}$.

   见书
3. 请编程实现 PTASKnapsack(I, k)。
4. 修改子集合问题为：找到某个子集，使得其元素和不小于 t，但又尽可能的接近 t。问如何修改算法 ApproxSubsetSum(S, t)来得到该问题的一个好的近似解。

   思路 1:

   1. L_0=Sum(x_i)

   2. L_i=L_{i-1}-x_i

   3. Remove

   思路 2:

   X 的元素可以全部取反-X, -t, 然后执行原算法

```
SubsetSum(S,t)
sum ← 'the sum of all values in S'
L0 ← sum
for i ← n to 1 do
      Li ← MergeLists(Li-1, Li-1 - xi)
      Li ← Trim(Li, ε/2n)
      remove from Li every element that is smaller than t
let z* be the smallest value in Ln
return z*
```

```
Trim(L, δ)
m ← |L|
L' ← y1
last ← y1
for i ←2 to m do
      if yi < last / (1 + δ) then
            append yi onto the end of L'
            last ← yi
return L'
```

在 trim 过程中每次选择用来代替的数都比被替代的数来得大。即如果当前的数（yi）可以用前一个选择了的数（last）替代则把当前的数（yi）选入已选择队列，并从已选择队列中删除 last。

```
Trim(L, δ)
m ← |L|
L' ← y1
```

```
last ← y1
for i ← 2 to m do
        if yi > last · (1 + δ)    then
                append yi onto the end of L'
                 last ← yi
        else
                remove last from L’ and append yi onto the end of L’
                   last← yi
return L'
```

习题 3.7

1. 最大割问题中，给予一个无权无向图 $G=(V, E)$。我们定义一个割 $C:(S, V-S)$和这个割的权，权值为这个割所含的边的数目。目标是找到一个最大权值的割。假定对于每个节点 v，我们随机且独立地以概率 1/2 将 v 置于 S 中，也以概率 1/2 将它置于 $V-S$ 中。证明这个算法是一个随机 2－近似的算法。

对于任意给定的一条边 $(u,v)\in E$，令随机变量 Y_{uv} 为

$$Y_{uv}=\begin{cases}1 & \text{如果}(u,v)\in C\\ 0 & \text{否则}\end{cases}$$

则，$|C|=\sum_{(u,v)\in E}Y_{uv}$，我们需要计算

$$E[|C|]=E[\sum_{(u,v)\in E}Y_{uv}]$$

利用期望的线性性质，可得

$$E[|C|]=E[\sum_{(u,v)\in E}Y_{uv}]=\sum_{(u,v)\in E}E[Y_{uv}]$$

由于 $E[Y_{uv}]=1\cdot\Pr\{Y_{uv}=1\}$

下面证明 $\Pr\{Y_{uv}=1\}=\frac{1}{2}$，注意到每个顶点都有同样的概率被选进或者不选进 S，即

$$\Pr\{Y_{uv}=1\}=\Pr\{(u\in S\text{且}v\in V-S)\text{或者}(u\in V-S\text{且}v\in S)\}$$
$$=\Pr\{u\in S]\Pr\{v\in V-S\}+\Pr\{u\in V-S\}\Pr\{v\in S\}$$
$$=\frac{1}{2}\times\frac{1}{2}+\frac{1}{2}\times\frac{1}{2}=\frac{1}{2}$$

因此 $E[|C|]=E[\sum_{(u,v)\in E}Y_{uv}]=\sum_{(u,v)\in E}E[Y_{uv}]=\frac{|E|}{2}$

又由于 $OPT(I)\le|E|$，由上两式可证。

6. 证明推论 3.7.1。

类似书上证明，略

习题 3.8

1. 证明推论 3.8.1。
类似书上证明，简单略
2. 请利用基于线性规划的随机近似算法求解 MAX-3-SAT，并分析其近似比。
类似书上证明，简单略

3. 给定一个完全图 $G=(V,E)$，以及对任意一条边 $e\in E$，指定一个权值 $w(e)>0$，$|V|$ 为偶数，最小权的完美匹配问题就是找到一个完美匹配 M 使得
$\min\sum_{e\in M} w(e)$.
请为该问题设计一个近似算法。
图 G 的一个完美匹配是 G 的一些相互独立的边的集合,并覆盖了 G 中所有的顶点。
整数规划算法:

$$z(e_{uv})=\begin{cases}1 & e_{uv}\in M\\ 0 & \text{否则}\end{cases}$$

目标函数为 $\min\sum_{e_{uv}} w(e_{uv})z(e_{uv})$

约束条件:

$$2\sum_{e_{uv}\in M} z(e_{uv})=|V|$$

$$\sum_{u\in V} z(e_{uv})=1$$

4. 请利用主对偶技术为集合覆盖问题设计一个近似算法。
令每个元素 x_i 对应一个变量 z_i，则对偶问题为

$$\max \quad \sum_{i=1}^{n} z_i$$

s.t.

$$\sum_{x_i\in S_j} z_i \le 1, j\in\{1,2,\cdots,m\}$$

$$0\le z_i$$

PrimalDual SetCover(X, F)

1 set $y_j=0$ for any S_j and $z_i=0$ for any element x_i

2 **repeat**

3 pick an uncovered element x_i, and increase y_j until some set becomes tight

```
        Add all newly tight sets to the cover C by setting y_j = 1 for those sets
until all elements are covered
return C
```

习题3.9

1. 证明如果 P≠NP 那么团问题不存在绝对近似算法 *A*。

   团是指 G 的一个完全子图,该子图不包含在任何其他的完全子图当中。

   完全子图：任意两点都相连的顶点的集合

   给定图$G=(V,E)$，我们可以如下构造图G^k：将图G复制k份（包括图G），不属于同一份的任意两个顶点，连接一条边。此时有 $\mathrm{OPT}(G^k)=k\,\mathrm{OPT}(G)$。

   下面可以反证法证明。假设团问题存在 k 绝对近似算法A，则有

   $$\left|A(G)-\mathrm{OPT}(G))\right|\le k$$

   则由上可以得到最优求解团问题的算法：

   对G^k运行算法A。如果图G中最大的团具有大小k，则有

   $$\left|A(G)-\mathrm{OPT}(G)\right|\le k \quad\Rightarrow\quad \left|A(G^{k+1})-(k+1)\mathrm{OPT}(G)\right|\le k$$

   不难看出，给定G^{k+1}中的任一个大小为S的团，我们可以在多项式时间内找到图G中大小为$S/(k+1)$，即$A(G)=\dfrac{A(G^{k+1})}{k+1}$。这样，我们能够找到图$G$中的团$A(G)$使得

   $$\left|A(G^{k+1})-(k+1)\mathrm{OPT}(G)\right|\le k \quad\Rightarrow\quad \left|(k+1)A(G)-(k+1)\mathrm{OPT}(G)\right|\le k$$

   整理上式可得

   $$\left|A(G)-\mathrm{OPT}(G)\right|\le \frac{k}{k+1}$$

   由于$A(G),\mathrm{OPT}(G)$均为整数，这表明$A(G)$一定是一个最优值，这就产生了矛盾。

2. 给定一个无向图G，令*U*为无向图*G*的顶点的子集，当且仅当对于*U*中的任意点*u*和*v*，(*u* , *v*) 不是*G*的一条边时，*U*定义了一个空子图。当且仅当一个子集*U*不被包含在一个更大的点集中时，该点集是图*G*的一个独立集（independent set)，同时它也定义了图*G*的空子图。最大独立集是具有最大顶点数的独立集。最大独立集问题是指寻找图*G*的一个最大独立集。对于该问题，存在一个绝对近似算法吗？如果存在，请设计出该算法，如果不存在，请证明之。

   假定存在k-绝对近似算法A，任给问题的一个实例G，有

   $$\left|A(G)-\mathrm{OPT}(G))\right|\le k$$

我们复制 G k份，再加上原图，可得一个新的图 G^{k+1}，因此OPT(G^{k+1})=(k+1)OPT(G)。

对图 G^{k+1} 运行A，同样有

$$\left|A(G^{k+1}) - \mathrm{OPT}(G^{k+1}))\right| \le k$$

同样有 $A(G^{k+1}) = (k+1)A(G)$ ，余下类似第一题的证明。

3. 证明如果 $R < \dfrac{7}{6}$，则对于顶点覆盖问题，不存在 $R-$近似算法。

4. 证明推论3.9.1对Bin Packing问题没有近似比小于3/2的近似算法，除非P = NP。
   与书上证明类似

5. 给出定理3.9.6的详细证明过程。

习题 3.10

1. 为例 2.1.2 聘任问题设计一个在线算法。
2. 为救护车调度问题设计一个随机在线算法。