2020SP

1. 问答题

1. 如果一个问题的理论上的最优时间复杂度为$\Omega(f(n))$，而一个解决该问题的算法的最坏时间复杂度为$O(f(n))$，则称该算法为最优算法。例如堆排序。
2. 解
   i. 由主定理得$T(n)=\Theta(n\log n)$
   ii. $a=1,b=\frac{4}{3},n^{\log_b a}=1,\exists c\leq 1,s.t.af(x/b)\leq cf(x)$,由主定理得$T(n)=\Theta(n)$
   iii. 直接展开后得$T(n)=O(n)$
3. 是，只需要证明最小生成树唯一即可，因为当最小生成树唯一时，由Kruscal算法得到的最小生成树便是唯一的最小生成树，即T，而$f(x)=x^2$递增，所以改变前后由Kruscal算法得到的最小生成树仍是T。所以T仍是最小生成树。事实上，可以用数学归纳法证明最小生成树唯一，按照图的顶点数量进行归纳。当$n=1$时，显然成立。假设当$n=k$时成立，当$n=k+1$时，任取一个顶点V，由归纳假设可得$G-V$的各个联通分量均有唯一的最小生成树，由于G为连通图，对于各个连通分量，有若干个顶点与V相邻，显然，在最小生成树中，只保留权重最小的那一条边，从而G的最小生成树也唯一。
4. 双指针

```
i = 1
j = n
while True:
    while A[i] 为奇数:
        i ++
    while A[j] 为偶数:
        j --
    if i>j:
        break
    Swap(A[i],A[j])
```

5. 将现实问题抽象，识别问题的特点并选用或设计相应的算法进行求解，同时还要尽可能进一步进行优化。
6. 采用分治法，选择一个枢纽元素，将数组内小于它的元素置于左侧，大于它的元素置于右侧，然后对其左右两侧的数组递归地使用快速排序。主要因素是枢纽元素的选取是否使得左右两侧数组的大小，即子问题的规模，差异小。可以将其随机化，即随机选取枢纽元素

7. 相同。二分检索利用分治法的思想，每次递归将原问题变成原先规模的一半，即依据和中间元素的比较结果，在原先一半的数组中继续检索。而最优二分检索树的思想也是分治法，依据和根节点的比较结果，进入左子树或右子树继续检索，问题规模变为原先的一半。$T(n)=T(n/2)+\Theta(1)$

2. 程序设计

3. 记分配给机器1的任务的指标集为I，停机时间为$T_1=\max\sum_I t_i,\sum_{\{1,\cdots,n\}-I}t_i$，机器2的停机时间$T_2=\sum_{\{1,\cdots,n\}-I}t_i$,记$\max\sum_I t_i,\sum_{\{1,\cdots,n\}-I}t_i$为T，则T为定值。由于对称性，不妨设$T_1\leq T_2$，则原问题等价于$\max T_1$，约束条件为$T_1\leq T//2$。使用动态规划进行求解。

$$dp=\begin{cases}\max\{dp[i-1][j],t_i+dp[i-1][j-t_i]\} & t_i\leq j\\ dp[i-1][j] & t_i>j\\ 0 & j=0或i=0\end{cases}$$

   时间复杂度和空间复杂度均为$O(Tn)$
4. 使用贪心算法

```
i = 1
j = 1
while i <= n and j <= m:
    if A[i] == B[j]:
        i ++
        j ++
    else:
        i ++
if j == m:
    return True
else:
    return False
```

贪心选择策略即S'中的元素在满足下标递增的约束下，总是与下标尽可能小的S中的元素匹配。

显然，若$a_{i_1},\cdots,a_{i_r}$与$b_1,\cdots,b_r$是S'在S中能匹配到的最长子序列，则$\forall 1\leq k<r$，$a_{i_k},\cdots,a_{i_r}$是$\{b_k,\cdots,b_m\}$在$\{a_{i_{k-1}+1},\cdots,a_n\}$中匹配到的最长子序列。从而有最优子结构性质。

下面对于证明按照该贪心选择策略，能够匹配到最长的子序列。

归纳基础：

假设i_0是满足$a_i = b_1$最小的数字，显然存在$i_1 = i_0$的最长子序列（任取一个最长子序列，若不满足，替换即可）。

假设当第k步时成立，则当$k+1$步时，最优子结构性质，剩余部分是子问题的一个最优解，而由归纳基础可得，按照贪心策略得到的第$k+1$步也包含在子问题的某个最优解中，从而按照贪心策略作出的$k+1$个匹配包含在某个最长子序列中。

从而该算法能够得出得到最长子序列（A[i]==B[j]时便视为对应元素匹配），因此，若匹配到的长度为m则说明S'是S的子序列，否则，说明不是，从而该算法正确。

5. 使用回溯法求解。
   解向量为5元组，$x_i \in \{0,1\}$
   解空间结构为解集树。
   约束函数是3个不等式