T1 – 麻将

解法：

$f[i][j][k]$ 表示目前考虑了大小为 $1,2,3,\ldots i$ 种麻将牌，构成若干面子，其中有j个$\{i-1,i,i+1\}$和k个$\{i,i+1,i+2\}$，且$1,2,\ldots,i$中的牌全部用完的方案数。

考虑大小为$i+1$的牌，假设$\{i+1,i+2,i+3\}$有q个，那么必然要满足$j+k+q\le a[i+1]$，且$(j+k+q) \bmod 3 = a[i+1] \bmod 3$，因为还可以放若干个$\{i+1,i+1,i+1\}$。

因此$f[i][j][k]$可以转移到$f[i+1][k][a[i+1]-j-k-3p]$其中p为任意非负整数，这样直接转移复杂度为$O(n^4)$。

可以先转移到$f[i+1][k][a[i+1]-j-k]$，再倒叙让所有$[i+1][k][q]+=f[i+1][k][q+3]$，这样就不用枚举$p$的具体大小了，复杂度为$O(n^3)$。

又因为$n\le 5000$，有效的$f[i][j][k]$状态不超过$O(n^2)$，因此复杂度为$O(n^2)$。

T2 – 序列

解法：

本题考察了构造的相关知识，可以直接上手构造，也可以通过观察小数据的规律输出结果。

30pts
$O(n!)$查找所有可行序列。

+30pts
$2k>n$一定无解，此时a_k无法填入任何数。
由于字典序最小，我们可以从前往后贪心地填。
有一个直接的想法，把每k个数分为一组，每次选能填的数中最小的。
按照这种方法，对于所有编号为奇数的组，我们填入$i+k$，否则填入$i-k$，即结果形如
$1+k,2+k,3+k,\ldots,2k,1,2,3,\ldots,k,3k+1,3k+2,\ldots,4k,2k+1,2k+2,\ldots,3k,\ldots$。
当n为$2k$的倍数时合法。

100pts
显然，这个构造在其他情况下是不一定合法的，我们需要调整，且希望改动尽量少的元素。考虑无解的条件，只需修改最后$2k$个即可。
观察发现，对于$i\in[n-2k+1,n-k]$这段a_i只能填$i+k$，否则会与前面冲突或导致后面$ai+k$处无数可填。
最后k个就把剩余的数从小到大填进去，易证必然合法且最优。

T3 – 芭蕾

解法：

30pts
排列暴力枚举能换就换进行爆搜，时间复杂度为$O(n!)$。

60pts
如果两个数$a_i + a_j > W$，那么它们的相对顺序永远不变。
如果整个序列所有数相加的和均小于等于W，那么所有序列都可以取到。
否则考虑取出整个序列最大的数a_i，对于剩下的数，如果加上a_i不超过W，那么说明其可以被换到整个序列的任意位置，我们用组合数将这个方案乘上，并把这些数从序列中删去，考虑剩下的数，左侧的数一定永远在a_i左侧，右侧的数一定永远在a_i右侧，将问题变成两个子问题递归即可，时间复杂度为$O(n^2)$。
考虑求字典序最小的解，把所有$a_i + a_j > W$连一条边，即为求字典序最小的拓扑排序，时间复杂度为$O(n^2)$。

100pts
先考虑求方案数，上述过程和笛卡尔树的形式类似，我们建出整棵树的笛卡尔树，通过倍增求解每个数具体是在哪个祖先处相加后小于等于W，时间复杂度为$O(n \log n)$。
再求字典序最小的方案数。大于$\frac{M}{2}$的数相对顺序不变，可以连一条链。对于不超过$\frac{M}{2}$的每个数a_i，找到左右第一个大于$M - a_i$的数，只需要它们之间的相对顺序不变即可（因为其他的顺序关系都通过链确定了），求解字典序最小的拓扑排序即可，之后用堆维护，求字典序最小的拓扑排序即可，时间复杂度为$O(n \log n)$ 。

T4 – 购票

解法：

主要考察了性质观察以及动态规划的相关内容。

50pts
考虑2^n枚举所有情况，如何计算最优策略下的答案。
我们令$f[i]$表示目前手上的车票截止时间为i时的最小花费。
如果$i + 1$时刻没有检票员，那么$f[i + 1] = f[i]$。
否则$f[i+1] = \min(f[i+1-75]+6, f[i+1-20]+2)$，即选择买两种车票中的一种，最终答案即为$f[200]$。
把上面的DP结果作为DP状态。
$f[i+1] = \min(f[i+1-75]+3, f[i+1-20]+1) \le f[i+1-75]+3$

100pts
两种车票的价格均为偶数，不妨先将票价除以2。
我们在计算时只关心$f[i], f[i-1], \ldots, f[i-74]$这 75 个值。
注意到两个性质：

1. f数组单调不降，即$f[i] \le f[i+1]$。
2. $f[i]$和$f[i-74]$相差不会超过3，因为$f[i] \le f[i+1] \le f[i-74]+3$。
   $f[i-74] = x$
   $f[i-74], f[i-73], f[i-72], \ldots, f[i]$

$x, x, x, x, x, x, x, 1+x, 1+x, 1+x, 1+x, 1+x, 2+x, 2+x, 2+x, 3+x, 3+x, 3+x, 3+x, 3+x$
a个x，b个$x+1$，c个$x+2$，$75-a-b-c$个$x+3$
75
5
80
只需要保留前 75
$S = (i, j, k)$
$dp[i][S]$下一步的乘务员和我当前的时间差了 t。

1. 下一个乘务员不来检查插入了t个0√
2. 下一个乘务员来检查插入$t-1$个0，做一次转移
   000000000000000111111111111112222222222222222333333333333
   000000000000000111111111111111111222222222222222222333333
   $f[i][S]->f[i+1][S_1],f[i+1][S_2]$
   $f[i+1][S_1]+=f[i][S],f[i+1][S_2]=f[i][S]$
   $\max(f[74]-3,f[19]-1)$。
   $\max(f[74]-3,f[19]-1)$
   $0,-1,-2,-3$。
   最后一位DP值增加了。
   -1000000000000000111111111112222222222222222222222222222
   0111111111111111111222222222233333333333333333333333333
   $-1:dp[i]->dp[i+1]$权值加了1。
   假设目前 dp 值为 0，$0+1=1$，$1->2$
   10
   $0->1,1->2,2->3,3->4,\dots,9->10$
   最终的答案是 10，最终的dp值是 10
   $0->1,1->2,2->3,3->4,9->10$
   加一个方案数。
   $f[i][S]$考虑了前i个乘务员的情况，目前的DP数组$f[i-74]-f[i]$的形态是S的方案数。
   2^{n-i-1}种可能。
   $f[i]\times 2^{n-i-1}$
   只保留前 75 位

因此$f[i-74]-f[i],f[i-74]-f[i-1],f[i-74]-f[i-2],\dots,f[i-74]-f[i-74]$这个数组至多只有$O(75^3)$种可能，即$i$个 3，j个 2，k个 1，$75-i-j-k$个 0，实际数量只会远小于75^3。

因此用$dp[i][S]$表示考虑了前i个时间点的乘务员状况，且$f[i]-f[i-75],f[i-1]-f[i-75],\dots$的状况为$S$的方案数，先预处理出每个$S$状态下下一个乘务员出现的时间间隔为$t$会转移到的状况，只需要处理$t\in[1,76]$，因为$t>76$时的答案是一样的，每次发现 DP 数组增加则乘上方案数，时间复杂度为$O(n\times 75^3)$。

30 分的部分主要是给实现不精确导致复杂度偏大的同学。