2.

双调巡游

可以先将所有节点按照横坐标从小到大排序，发现除去最左结点A进而最右结点B，将剩下结点分成两组，向左巡游即是从A出发，经过其中一组(按横坐标升序)， 到达B， 向右巡游就是从B出发，经过另一组(按横坐标降序)， 到达A， 所以对于每个点(除去端点)，都可以分到两个组。

所以可以从最左第二个点到最右第二个点进行dp, 当讨论到第i个结点的时候，有一组的最后一个节点肯定是i - 1, 有一组的最后一个结点肯定是$j(1 \leq j < i-1)$， 此时所有连线长度可以记做$dp(i-1,j)$, 若是第i个节点与i- 1个结点相连，则可以去更新$dp(i,j)$, 若是第i个结点与第j个结点相连，则可以更新$dp(i,i-1)$

可以得到递推式:

$$dp(i,i-1) = min\ dp(i-1,j) + dis(i-1,j)\ (j < i-1)$$
$$dp(i,j) = dp(i-1,j) + dis(i,i-1)$$

此时时间复杂度为 $O(n^2)$, 由于计算每个点间的距离就需要 $O(n^2)$ 的时间，所以这应该是较好的复杂度了。

空间复杂度若是采用滚动数组可以从$O(n^2)$降到$O(n)$

整齐打印

第一个单词只能放在一行的开头，而后面的单词，能放在前面一个单词的后面(如果放的了的话)，或者另起一行来放。

所以可以记录前一个单词放好以后有哪些情况(一种情况保留两个信息，一个是最后一行还剩几个空白，一个是这种情况除最后一行额外空格的立方之和)。

理论上每增加一种单词，会多出一种情况(这个单词单独占有一行的情况)， 但是也很有可能其他的情况后面不能摆这个单词，只能另起一行，这样情况数就会减少，因此最坏复杂度为 $O(n^2)$， 大部分情况应该是到不了这个上界的

空间复杂度若是采用滚动数组可以从$O(n^2)$降到$O(n)$

斐波那契数列和

一个数是可以用斐波拉契数列计数的，如$16 = 1 * fib(6) + 1 * fib(3) (fib(6) = 13, fib(3) = 3)$, 因为我可以每次剪去 小于等于这个数 的最大一个斐波那契数列项，直到剪到等于0(若不等于0表示还可以继续剪)

而将斐波那契数列的某一项拆分的话， A_i可以拆成$\{A_{i-1}, A_{i-2}\}$, A_{i-1} 是不可以继续拆分的(因为拆分A_{i-1}必然还会再出现一个A_{i-2})， 但可以拆分A_{i-2}, 则可以继续拆分成$\{A_{i-1}, A_{i-3}, A_{i-4}\}$, 可以推出第i个斐波那契数列可以有$(i-1)/2$种拆法

所以一个数可以拆成斐波那契数列计数(每次减去最大的能减去的斐波那契数)， 从大到小分别是$A_0, A_1, A_2 \ldots$, 分别是第$pos_0, pos_1, pos_2 \ldots$个斐波拉契数， 若是$A_1$拆分，则可以将$A_0$在$[A_1, A_0]$范围里拆分(若是拆分到小于$A_1$的话，可以证明与$A_1$的拆分会重复), 若是$A_1$不拆分，则可以将$A_0$在$(A_1, A_0]$范围里拆分

可以得到递推式：（0表示不拆分，1表示拆分）

$$dp[i][0] = dp[i+1][0] + dp[i+1][1]$$
$$dp[i][1] = dp[i+1][0] \times (pos[i] - pos[i+1] - 1)/2 \ + \ dp[i+1][0] \times (pos[i] - pos[i+1] - 1)/2$$

假设小于等于$1e18$的斐波那契数列共有c项(c应该是稍大于64的一个常数)， 则一个数可以拆成斐波那契数列计数需要$O(c)$时间，dp过程需要$O(2c)$时间，则一次需要的是常数时间，T次询问需要$O(cT)$

3.

a. 可以记dp(i, j) 表示X串遍历到第i个元素，Z串遍历到第j个元素，而我们所要求的就是dp(m, n)

根据六种操作，可以得出以下递推关系式：

$$dp[i][j] = min \begin{cases} dp[i-1][j-1] + cost(copy) \quad if\ x[i] = y[j] \\ dp[i-1][j-1] + cost(replace) \quad if\ x[i] \neq y[j] \\ dp[i-2][j-2] + cost(twiddle) \quad if\ x[i] = y[j-1]\ \&\&\ x[i-1] = y[j] \\ dp[i-1][j] + cost(delete) \\ dp[i][j-1] + cost(insert) \end{cases}$$
$$dp[m][n] = min(dp[i][n] + cost(kill))$$

时间空间复杂度为$O(mn)$

b. 可以进行类比，若是两个位置相同，则相当于copy, 两个位置不同(且都不是空格)， 则相当于replace, 若两个位置有一个是空格，则是delete或者是insert

由于编辑距离问题计算的是最小值，所以可以让copy数值为 -1， replace为1， delete或者insert为2，不用kill 和 twiddle(可以让这两种操作cost为INF), 计算得到的最小值的相反数，就是DNA问题的分数。