Solution: Homework 2

B. 2-SAT

输出方案时，注意到拓扑序大 (如 $p \to q$ 中的 q) 的更容易被满足，因此输出 $x_i = 0$ 和 $x_i = 1$ 中拓扑序大的即可，因为不存在一条边从拓扑序大的指向小的。

本题数据采用的是 OJ 上其他 2-SAT 的数据，因此本题存在 $O(nm)$ 的暴力过了的情况，不过很少，所以就没改数据重测了。

C. Shortest path

用任意方法实现单源最短路即可。除了 Floyd，应该没有什么算法能 TLE。

D. 0-1 Shortest path

0-1 bfs。bfs 的时候遇到 0 边 push front，1 边 push back。

由于输入大小其实是 $m \log 10^9$ 的，因此主要复杂度其实在输入数据上，所以用堆优化 Dijkstra 可以水过去。

当然，队列优化的 Bellman Ford (a.k.a. SPFA) 是过不去的，因为其最坏复杂度 $O(nm)$。而某些采用 Small Label First 的 SPFA 本质上就是该题的正确做法。

E. Negative cycle

输出方案时，直接从一个 Bellman Ford 额外轮被更新的点找前驱就行了。由于到第 n 轮仍然被更新，因此至少往前 n 步都是有前驱的，因此往前 n 步一定在环上。

当然，注意到可能这个点的前驱被更新过，可能出现类似 1, 2, 3, 4, 3, 4 ... 的情况，所以最好是往回找到第一个重复的，然后把这一段拿出来作为负环。

队列优化的 Bellman Ford (a.k.a. SPFA) 复杂度亦是正确的，此时判断条件为入队 $n+1$ 次。

栈优化的 Bellman Ford (a.k.a. dfs-SPFA) 是过不去的，因为其最坏复杂度 $O(2^n)$。

F. Strict k-th shortest path

本题尝试出成一个需要比较多思考的题，但是存在一个简单贪心做法导致难度变低了很多。

sol -1.

$O(k \log k)$ 的非严格 k 短路算法（即，相同权重算多条的 k 短路算法）是无法通过这个题的，因为可能存在 $\binom{m}{k-1}$ 条严格 k 短路。采用类似 SPFA 或者 A* 的迭代亦是如此。

sol1.

跑 Dijkstra 时，每个点出堆前 k 次（正常 Dijkstra 是 1 次）都往后做一次更新。此时，每个点前 k 次出堆即为到这个点的前 k 严格短距离。参考 sol2 的结论，这个做法易证是对的。复杂度 $O(km \log(km))$。

sol2.

Definition:

- 对于一条边 $e=(u,v,w)$，我们定义 $\Delta_e = w+\operatorname{dis}(v,t)-\operatorname{dis}(u,t)$ ，同理可以对路径定义。
- 我们称所有经过后会导致到终点的最短路上升的边为差边，也即 $\Delta>0$ 的边；其余 $\Delta=0$ 的边称为好边。

按定义可知，s 到 t 中不经过任何差边的路径 $\Delta=0$，为最短路。我们可以将这个结论推广到 k 短路上。

Lemma:

- 任何一条严格 k 短路上最多有 $k-1$ 条差边。

证明：否则一定能找到从起点出发，保留前 $0,1,2,3,\ldots,k-1$ 条差边，而分别在这之后走最短路，使得有 k 条长度两两不同的路径比考虑的路径短。

类似 Bellman Ford，考虑一条一条把差边找出来，且由于最多只有 $k-1$ 条差边，因此最多找 $k-1$ 次。

与此同时，我们可以发现这个路上从后往前一条一条把差边删掉，得到的路径必然是某个 $j<k$ 短路；并且这些差边之间是用最短路连接的。

因此，我们迭代 $k-1$ 轮，在第 i 轮时从 i 短路（设其差值为 Δ_i）出发尝试更新所有点。

最直观的做法是：对于一条坏边 $e=(u,v,w)$ 来说，如果能够更新，需要满足存在一条 $s\to u$ 的路径 Δ_p，使得 $\Delta_p=\Delta_i$ （因此 e 接在这条路后面能成为最后一条坏边）；在更新完后，所有在 v 到 t 最短路上的点 x，都能得到一条 $s\to x$ 的 $\Delta_i+\Delta_e$ 路径。

如果直接暴力按照上面的做法更新，每个点直接维护到这个点的前 k 小 Δ 的集合，然后每次把没有考虑过的最小的定为 $i+1$ 短路，就可以正确解决这个题了。但是，由于 v 到 t 的最短路径上的点可能非常多，复杂度是无法接受的。

考虑进行优化。如果不考虑零环的话，所有的好边构成一个有向无环图，有向无环图按照拓扑序可以直接将所有拓扑序低的信息传递给高的。因此，我们把上述更新拆为两步：

- 对于每个含有 Δ_i 的点，往后继更新一次。此时，$\Delta_i+\Delta_e$ 的信息在 v 上，还没有传递给 v 到 t 的最短路上。
- 此时所有集合中出现过的比 Δ_i 严格大的最小值必然是 $i+1$ 短路的差值 Δ_{i+1}。按照拓扑序，把所有含有 Δ_{i+1} 信息的点向后传递。这些点也即下一轮将往后更新的。

这样的话复杂度也降到了 $O(k^2m)$：一共 k 轮，每轮最多传递 $2m$ 次信息，维护集合即使采用插入排序暴力维护也是最多 k 步。

现在考虑零环，零边构成的强连通分量里的点的距离都是相同的，因此将每条边的端点替换为强连通分量的标号即可。

因此，完整的算法过程如下：

1. 对 0 边建图跑强连通分量缩点；
2. 从 t 出发跑单源最短路，并求出所有在 s 到 t 最短路上的点，往这些点维护的差值集合中插入 $\Delta_1=0$；
3. 保留所有好边，在图上跑拓扑排序；
4. 迭代 $k-1$ 轮，每轮尝试用 i 短路的差值 Δ_i 来尝试更新可能出现的 $\Delta_{i+1},\ldots,\Delta_k$ ：
   - 对于每个含有 Δ_i 的点，往后继更新一次。
   - 此时所有集合中出现过的比 Δ_i 严格大的最小值必然是 $i+1$ 短路的差值 Δ_{i+1}。
   - 按照拓扑序，把所有含有 Δ_{i+1} 信息的点向后传递。

采用数据结构的最优复杂度是 $O(mk\log k+m\log m)$。实际上来说由于 k 较小，插入排序维护集合的 $O(mk^2+m\log m)$ 是更合适的实现。

如果这个题只有这个做法就好了，因为是用一个非常有趣的观察，继承 Bellman Ford 的思想，采用了 Tarjan, Dijkstra 和拓扑排序，最终解决了本题。

噢，但是非常遗憾，km 大小堆的 Dijkstra 跑得实在太快了。