1. Because of the recent droughts, N proposals have been made to dam the Murray river. The i^{th} proposal asks to place a dam x_i meters from the head of the river (i.e., from the source of the river) and requires that there is not another dam within r_i metres (upstream or downstream). What is the largest number of dams that can be built? You may assume that $x_i < x_{i+1}$.

   Solution *Consider the following subproblems $P(i)$ for all i such that $1 \leq i \leq n$:*

   $P(i)$: find a subset B_i of the set $S_i = \{x_1, \ldots, x_i\}$ of positions where to build dams so that position x_i is included in B_i and B_i has the largest possible number of elements satisfying the conditions of the problem.

   Note first that for all $i < N$ if for some j position x_j satisfies $x_j < x_i - r_i$, then this is also true for all $k < j$. Similarly, for all k such that $k < j < i$ and which satisfy $x_k + r_k < x_j$ then also $x_k + r_k < x_i$. Let B_j be optimal choice of positions where to build dams up to and including position x_j. To obtain an optimal solution for S_i it is enough to look for all j such that $x_j + r_j < x_i$ and $x_i - r_i > x_j$ and among such j pick one with the largest number of elements in the corresponding solution B_j and let $B_i = B_j \cup \{x_i\}$. IMPORTANT: After you solve all these problem, optimal solution to the original problem need not be the solution for problem $P(n)$; now from all solutions for $P(1), \ldots, P(n)$ you have to pick the one with the largest number of elements. Note that we forced B_i to include x_i and so $x_N \in B_N$ but the optimal solution for the original problem might not include X_N.

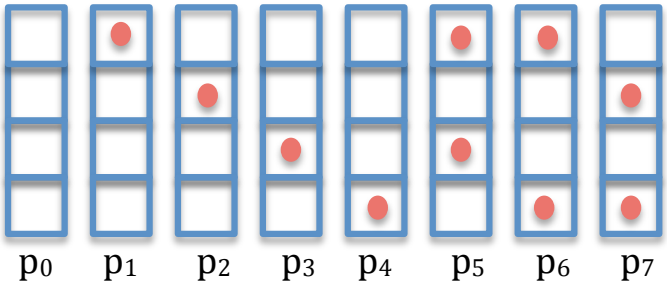
2. We are given a checkerboard which has 4 rows and n columns, and has an integer written in each square. We are also given a set of $2n$ pebbles, and we want to place some or all of these on the checkerboard (each pebble can be placed on exactly one square) so as to maximize the sum of the integers in the squares that are covered by pebbles. There is one constraint: for a placement of pebbles to be legal, no two of them can be on horizontally or vertically adjacent squares (diagonal adjacency is fine).

   (a) Determine the number of legal *patterns* that can occur in any column (in isolation, ignoring the pebbles in adjacent columns) and describe these patterns.

   Call two patterns *compatible* if they can be placed on adjacent columns to form a legal placement. Let us consider sub-problems consisting of the first k columns $1 \leq k \leq n$. Each sub-problem can be assigned a type, which is the pattern occurring in the last column.

   (b) Using the notions of compatibility and type, give an O(n)-time algorithm for computing an optimal placement.

   Solution *Finding all legitimate column patterns is easy; since no two pebbles can be adjacent, there can be at most two pebbles in each column. All possibilities are shown below, enumerated 0 to 7:*



We now find for each column pattern p_i what column patterns are compatible as an adjacent column; note that this relation is symmetric: if p_i is compatible with p_j as an adjacent column pattern on its left, then p_j is also compatible with p_i as an adjacent column pattern on its left:

Table 1: Compatibility relation $\text{comp}(p_i, p_j)$

	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7
p_0	1	1	1	1	1	1	1	1
p_1	1	0	1	1	1	0	0	1
p_2	1	1	0	1	1	1	1	0
p_3	1	1	1	0	1	0	1	1
p_4	1	1	1	1	0	1	0	0
p_5	1	0	1	0	1	0	0	1
p_6	1	0	1	1	0	0	0	0
p_7	1	1	0	1	0	1	0	0

Let the number in cell (i,k), $1 \le i \le n$, $1 \le k \le 4$ be $n(i,k)$ and for all $1 \le i \le n$ and all $0 \le j \le 7$ consider a subproblem $P(i,j)$: "Find optimal solution for checkerboard with i columns, such that the last, i^{th} column has column pattern p_j".

Recursion: Let $\text{opt}(i,j)$ *be the optimal solution for sub-problem $P(i,j)$. then*

$$\text{opt}(i,j) = \max_{m:\text{comp}(p_m,p_j)} \text{opt}(i-1,m) + \sum_k \{n(i,k) : \text{cell } (i,k) \text{ is covered by a pebble from } p_j\}$$

After you find $\text{opt}(n,j)$ for all $0 \le j \le 7$ pick among such solutions one with the largest score.

3. Skiers go fastest with skis whose length is about their height. Your team consists of n members, with heights $h_1, h_2, \ldots, h_n$. Your team gets a delivery of $m \ge n$ pairs of skis, with lengths $l_1, l_2, \ldots, l_m$. Your goal is to write an algorithm to assign to each skier one pair of skis to minimize the sum of the absolute differences between the height h_i of the skier and the length of the corresponding ski he got, i.e., to minimize

$$\sum_{1 \le i \le n} |h_i - l_{j(i)}|$$

where $l_{j(i)}$ is the length of the ski assigned to the skier of height h_i.

Solution *Order all skiers S_i, $1 \le i \le n$ by increasing height $h(S_i)$ and all skis s_j, $1 \le j \le m$, by increasing length $l(s_j)$. Now notice that if an assignment is optimal and $h(S_i) < h(S_j)$*

then the skis assigned to skier S_i can be taken shorter than the skis assigned to skier S_j, otherwise you could swap their skis without increasing the sum of the absolute values of the differences between the heights of the two skiers and lengths of their two pairs of skis. To see this, there are 6 cases to consider in terms of relative "sizes" of two skiers $h(S_1) < h(S_2)$ and two pairs of skis of "sizes" $l(s_1) < l(s_2)$:

$$h(S_1) \le h(S_2) \le l(s_1) \le l(s_2) \tag{0.1}$$
$$h(S_1) \le l(s_1) \le h(S_2) \le l(s_2) \tag{0.2}$$
$$h(S_1) \le l(s_1) \le l(s_2) \le h(S_2) \tag{0.3}$$
$$l(s_1) \le h(S_1) \le h(S_2) \le l(s_2) \tag{0.4}$$
$$l(s_1) \le h(S_1) \le l(s_2) \le h(S_2) \tag{0.5}$$
$$l(s_1) \le l(s_2) \le h(S_1) \le h(S_2) \tag{0.6}$$

Then it is easy to see that assigning longer skis to the shorter skier results in the sum $|h(S_1) - l(s_2)| + |h(S_2) - l(s_1)|$ which is larger than the sum $|h(S_1) - l(s_1)| + |h(S_2) - l(s_2)|$ in cases 2-5 and in cases 1 and 6 equal to that sum. (Draw these values on a number axis and compare the two sums of the lengths of the appropriate segments).

Consider now the following subproblems $P(i, j)$ for all i and j satisfying $1 \le i \le n$ and $i \le j \le m$:

$P(i, j)$:"Find optimal assignment of the first i skiers to skis which are among the first j skis."

If $j = i$ then there is only one assignment that assigns skis according to the skier's height resulting in

$$\text{opt}(i, i) = \sum_{k=1}^{i} |h(S_k) - l(s_k)|$$

If $j > i$ then there are two choices: either you assign to the i^{th} skier skis j or you do not. Thus, for $j > i$

$$\text{opt}(i, j) = \min(|h(S_i) - l(s_j)| + \text{opt}(i - 1, j - 1),\ \text{opt}(i, j - 1))$$

The final solution is $\text{opt}(n, m)$.

4. You know that $n+2$ spies S, $s_1, s_2, \ldots, s_n$ and T are communicating through certain number of communication channels; in fact, for each i and each j you know if there is a channel through which spy s_i can send a secret message to spy s_j or if there is no such a channel (i.e., you know what the graph with spies as vertices and communication channels as edges looks like).

   (a) Your task is to design an algorithm which finds the fewest number of channels which you need to compromise (for example, by placing a listening device on that channel) so that spy S cannot send a message to spy T through a sequence of intermediary spies without the message being passed through at least one compromised channel.

   (b) Assume now that you cannot compromise channels because they are encrypted, so the only thing you can do is bribe some of the spies. Design an algorithm which finds the smallest number of spies which you need to bribe so that S cannot send a message to T without the message going through at least one of the bribed spies as an intermediary.

Solution *a) is a straightforward max flow/min cut on a network with spies as vertices, with S as the source and T as the sink and with the capacity of all edges (communication links*

between spies) set to 1. After running a max flow algorithm obtain the resulting min cut; the edges (communication links) across such a cut have to be compromised and the number of such edges is equal to the capacity of such a min cut.

For b) Replace each spy vertex s_i (except S and T) with two duplicate vertices s_i^{in} and s_i^{out} and add an edge of a unit capacity from s_i^{in} to s_i^{out}; redirect all incoming edges of s_i to s_i^{in} and all outgoing edges from s_i will now be going out of s_i^{out}. The capacities of all other edges are set to infinity. Now use Edmonds-Karp algorithm to find max flow in such a network and find an associated min cut as the set of vertices still reachable in the last residual network flow. Find the edges crossing the cut. If they are all of capacity 1 they correspond to the spies who needs to be bribed; otherwise the algorithm returns output "No solution - S can communicate with T by a direct channel."

5. You are given a flow network G with $n > 4$ vertices. Besides the source s and the sink t, you are also given two other special vertices u and v belonging to G. Describe an algorithm which finds a cut of the smallest possible capacity among all cuts in which vertex u is at the same side of the cut as the source s and vertex v is at the same side as sink t.

   Solution *Just add an edge from s to u and an edge from v to t, both edges with infinite capacity and find min cut in such a graph. Another way of doing it is by adding a super source S connecting it with an edge of infinite capacity to the original source s and connecting it with another edge of infinite capacity with vertex u and also adding a super sink T connecting the original sink t and vertex v with T via two edges of infinite capacities. Now find a min cut in such a flow network.*