Name	Types	Important Insights	Programming tricks	Exam Year
Burning Coins from Two Sides	Dynamic Programming	Take the max of the min to get guaranteed amount		
Corbusier	Dynamic Programming	DP with nr_disks * k. For each disk, either take it or don't.		
Defensive Line	Dynamic Programming	DP with nr_defenders * nr_attackers. Do preprocessing on the possible choices, then use recursion to solve it		17
Magician and the Coin	Dynamic Programming	Each time we can bet a value between 1 and k, we want to find the maximum of that. If we manage to get enough, the probability is 1. If we do not manage, the probability is 0.		
Poker Chips	Dynamic Programming	-	map.find instead of find(map) maps much slower than vector for memo	
Punch	Dynamic Programming	DP with nr_beverages * maxVolume size		16
San Francisco	Dynamic Programming	DP with nr_holes * nr_moves. Do not actually need graph, just info where you can go from u. Solve recursively what score can be achieved with i many moves left. Do linear search over this variable.		
The Great Game	Dynamic Programming	Realize that the two marbels are independent of each other. In each move, it is tried to minimize, then maximize the number of turns. In the end decide based on the number of moves needed.		
New Tiles	Dynamic Programming & Bitmagic	DP with h * (2^w). Use bitmask to fill table in a very strange way		
Almost Antenna	Geometric	Construct the min circle, then go over support points and try without one of these.		
Antenna	Geometric	Straight forward, construct min circle.		
First Hit	Geometric	Instead of a ray, use a segment to check for intersection that keeps getting shorter (then we have to do less constructions). To make sure that we do not get an adversarial input, we do a random_shuffle before.		
Hit	Geometric	Straight forward using a ray and do_intersect		
Important Bridges	Graph: Biconnected Components	Realize the fact that bridges are biconnected components with only one edge. Use the template from the boost documentation		

New York	Graph: DFS	Do a DFS over the tree while always keeping the whole path saved (with the recursive implementation). Keep a multiset of the currently relevant temperatures, always check if max - min is below threshold.	rbegin(), rend(). Use global variables if the recursive DFS causes a stackoverflow.	18
Evolution	Graph: DFS & binary search	Do bfs over graph and handle on the fly all requests, since they can be computed in a different order, then later put back in the correct order.		16
Bobs Burden	Graph: Dijkstra Shortest Path	Undirected graph with vertex weights. Transform into directed graph with edge weights. Do Dijkstra from all three corners. Find best center by combining all three shortest paths		
Buddy Selection	Graph: Maximum Cardinality Matching	Realize that we do not need to compute the matching, but only test if there is a better one. For all student pairs, find the number of common hobbies by using set intersection. Then build a graph with edges only if the two students have more than f common hobbies. Check if there exists a maximal matching in this graph. If yes, it was not optimal, if no, it was optimal.		
Her Majesty's Secret Service	Graph: Maximum Cardinality Matching	Quite hard to see the matching part. Binary search over possible end times.		
Return of the Jedi	Graph: MST & BFS	Find the second minimum spanning tree cost. Compute MST, then compute for each pair of vertices the maximum edge in the spanning tree. Finally, find the cheapest edge to add that is not in the minimum spanning tree that could be added instead of the maximum edge in the path between these vertices.		14
Ant Challenge	Graph: MST & Dijkstra	Figure out that we need MST, then build new graph with MSTs		
Planet Express	Graph: Strong Components & Dijkstra	Use strong components to find all teleportation networks. Then connect them all by adding a vertex for each component and connecting all vertices of the component to it in both directions. Then use dijkstra to find the best warehouse.		

Attack of the Clones	Greedy	Circular earliest deadline first scheduling. Special input restrictions can be used to speed up the process		14
Boats	Greedy	Do earlierst deadline first scheduling		
Dominoes	Greedy	Iterate over it once to find the first domino that does not fall.		
Moving Books	Greedy	Binary search over solution space	have two indices and while(i < max && j < max)	
Octopussy	Greedy	Realize that it is greedy with respect to the minimum time on top of current. Realize that sorting will keep the invariant that a bomb on top has a higher number than on the bottom.		15
Diets	Linear Programming	Straight forward		
Radiation	Linear Programming	Loop over degrees to get all possible combinations		
The Empire Strikes Back	Linear Programming & Triangulation	Realize that radius should be maximal for sure, calculate using triangulation. Then do linear program over the power constraint		
World Cup	Linear Programming & Triangulation	Realize that it is not possible with min cost max flow. Use triangulation and set difference to do preprocessing on profits. Then regular linear program		17
Cantonal Courier	Max Flow	Weird flow problem. It does not matter who pays, but only that someone does (flow)		
Coin Tossing Tournament	Max Flow	For each player have a node. For every game that was not recorded, add an edge. Limit the flow from the player to target with the remaing points needed. Check if the flow is maximal.		
India	Max Flow	Limit maxflow to value, find the minimal cost. Do binary search over solution space.		18
Kingdom Defense	Max Flow	The vertex demands and supply can be easily modelled by adding a source and target vertex. For every edge (u,v) we know that at least c many units need to flow and at most C, i.e. we can add an edge from s to v with capacity c, from u to t with capacity c and from u to v with capacity C - c. Then we check if the maxflow is >= sum demands + sum miniums		
London	Max Flow	Create back/front pairs (a counter for each). For each of them have a node in the graph with a limit on them (have to choice either one of the other side)	letter to int with (int) letter - (int) 'a'	18

Marathon	Max Flow	Do a lot of preprocessing: eliminate duplicate edges by only taking shortest one and summing up capacities. Find all shortest paths by doing dijkstra from both sides, then iterating over all edges.		17
Shopping Trip	Max Flow	Find the number of edge disjoint paths, check by doing a max flow with capacity 1. Check if the max flow is equal to the given number of stores		
Tetris	Max Flow	Check corner cases	Number of (unconnected) nodes heavily impacts performance of push_relabel_max_flow()	
Satellites	Max Flow: Bipartite Minimum Vertex Cover	Compute flow, then residual BFS and read off result		
Algocoon Group	Max Flow: Minimum Cut	First find out which is the start and end point by trying every startpoint (calling push relabel max flow multiple times on the same graph is not slow!). Then read off the figures by doing a residual BFS.	Calling max-flow multiple times is totally fine. You can look at the runtime of push_relabel as roughly O(n^2) but that is no guarantee.	
Canteen	Min Cost Max Flow	Classical Min Cost Max Flow		15
Carsharing	Min Cost Max Flow	Ensure maximal flow by having edges with 0 cost. Make edge weights positive, use "path compression" to have only times that are relevant		15
Casino Royale	Min Cost Max Flow	Relatively standard. Have ensured max flow.		
Fleetrace	Min Cost Max Flow	Straight forward		17
Real Estate	Min Cost Max Flow	Classical Min Cost Max Flow		15
Highschool Teams	Split & List	-	Find range that satisfies by using equal_range, then subtract the two iterators	17
Light at the Museum	Split & List	-		16
Planks	Split & List		-	16
Planks	Split & List	Split into 4 sets, then regular split and list with lower and upper bound. Realize that there are 4! = 24 possibilities to label a 4-tuple.		
Even Matrices	STL: Partial Sum	Magic	-	15

Even Pairs	STL: Prefix Sum	Use prefix sum to avoid computing the sum of a sequence. Speed up even more by using a magic formula		
Deck of Cards	STL: Prefix Sum & Binary Search	Use binary search to get two best candidates for the sum in the prefix vector		16
Beach Bars	STL: Sliding Window	-		
Hiking Maps	STL: Sliding Window	For each segment find which path parts are inside. Then do a sliding window over the parts to find the cheapest segment that covers the whole path.	Use CGAL::left_turn(), CGAL::right_turn() to find if a point is inside a triangle	
Search Snippets	STL: Sliding Window	Slide over the sequence always updating the word counts and saving the shortest sequence		
Light Pattern	STL: Sliding Window & Bitmagic	Go from back to front. Always keep a count if the bits are inverted or not. Check all cases of odd and even swaps.		
Bistro	Triangulation	Straight forward		
Clues	Triangulation	Weird graph properties. Can do 2-coloring greedily	Can use struct as vertex info	
Germs	Triangulation	Realize that it only depends on the closest distance to any other germ or the boundary, hence use triangulation.		
Graypes	Triangulation	Straight forward, shortest edge is in triangulation		
H1N1	Triangulation		-	
Light at the Stage	Triangulation	Compute then each participant gets hit. If some do not get hit, output them, else get the ones that are hit last.		
Hong Kong	Triangulation & MST & BFS	Realize that all escape points at the centers of the triangles, build graph with all possible paths and from escape points to infinite vertex. Then do a minimum spanning tree and a BFS over it to find for each location if escape is possible	Use squared_radius(x, y, z) to find the squared distance of the circle that x, y and z span.	18
Golden Eye	Triangulation & Union Find			