Exercise 1.

We can find the specified matching with the help of the following algorithm. We initialize our new matching M with the intersection of M_1 and M_2. To find the remaining edges that should be in M, we look at all continuous paths in the symmetic difference between M_1 and M_2. For these paths, there are two possibilities, they either have an even length or an odd one. If a given path has an odd length, we take every second edge in the path (the edges which come from the matching that contributes more edges) and add them to M. If the path has an even length, if the path starts and end in V_1, we add all edges in p that are also in M_1 to M, otherwise we add all the edges which are in p and in M_2.

The following pseudocode describes this algorithm more formaly.

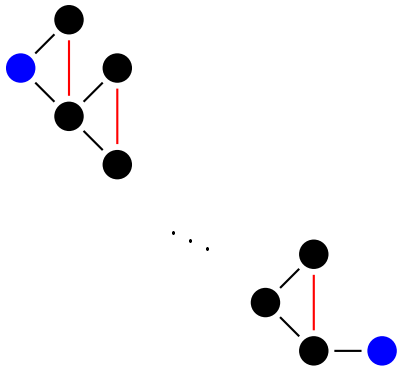
Algorithm 1: An algorithm to find the specified matching

Result: Find a matching M that matches all the vertices in V_1 that are matched by M_1 and all the vertices in V_2 that are matched by M_2

```
1  M = M_1 ∩ M_2;
2  foreach p ∈ all continuous paths in M_1 ⊕ M_2 do
3  |  if |p| is odd then
4  |  |  if |M_1 ∩ p| > |M_2 ∩ p| then
5  |  |  |  M = M ∪ (M_1 ∩ p);
6  |  |  else
7  |  |  |  M = M ∪ (M_2 ∩ p);
8  |  |  end
9  |  else
10 |  |  if p starts and ends in V_1 then
11 |  |  |  M = M ∪ (M_1 ∩ p);
12 |  |  else
13 |  |  |  M = M ∪ (M_2 ∩ p);
14 |  |  end
15 |  end
16 end
```

Exercise 2.

a) Consider the following graph:



If BFS is started from the free vertex on the top left, basically $\frac{n}{3}$ and so $\mathcal{O}(n)$ blossoms need to be shrunken down until the free vertex on the bottom right is reached.

b) In the algorithm presented in the lecture, the BFS is always restarted from all free vertices of the residual graph $G' = G \setminus B$ if a blossom was found and shrunken. since the path, which was already found until the base of the blossom was reached does not change, the BFS can be continued from the state which it was in when the base of the blossom was reached. The algorithm can be altered in the following way:

```
   Result: maximal matching M from a general graph G
 1 M ← ∅
 2 while ∃ augmenting path do
 3 |   Start BFS parallel from all free vertices in G
 4 |   if Blossom B found then
 5 |   |   G' ← Shrink B to b in G
 6 |   |   Alter all BFSs which include vertices from B with respect to b so that
 |   |     the new BFS' are semantically equivalent in G' relative to the original
 |   |     BFS in G
 7 |   |   Continue with the BFS'
 8 |   end
 9 |   if Augmenting path P found then
10 |   |   M ← M ⊕ P
11 |   |   Unfold B back according to the BFS associated
12 |   end
13 end
```

Runtime:

- one BFS takes $\mathcal{O}(n+m)$
- one shrink takes $\mathcal{O}(m)$ therefore BFS $\rightarrow$ BFS' takes in $\mathcal{O}(m)$
- still $\mathcal{O}(n)$ shrinkings in worst case
- Between two augmentations the runtime is therefore $\mathcal{O}(n+m)$
- Unfolding takes place in $\mathcal{O}(n+m)$ since it is based on the BFS that found the blossom

In total, one iteration is running in $\mathcal{O}(n+m)$. Since there are up to $\mathcal{O}(n)$ augmentations, the runtime in total is $\mathcal{O}(n \cdot m)$

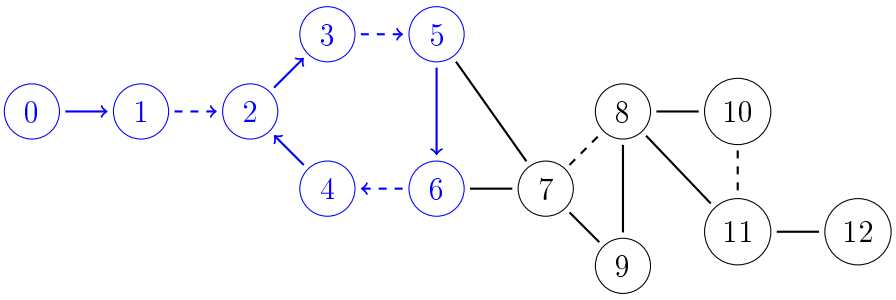
Exercise 3.

Since G_R contains an edge between all tasks that are not connected by a directed path in D, an edge (t_i, t_j) in G_R represents that the two tasks t_i and t_j are independent and could therefore be executed in parallel. A matching in G_R can then be understood as groupings of two tasks that will are scheduled at the same time (i.e. run in parallel on the two machines).

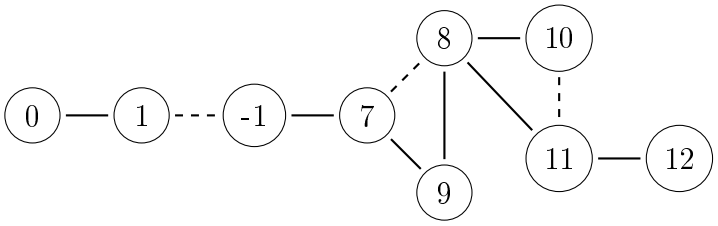
If we consider the maximum matching M, then G_R tells us which tasks will run without a "partner task" (i.e. the free vertices with respect to M) and which tasks are run in parallel with what other task. Since M is maximum, we can run no more tasks on the second machine, because of the dependencies. If we look at the runtime, the worst scheduling would have a total runtime of $\tau^*_{\text{worst}} = |T|$ (no tasks executed in parallel) and for each two tasks we match, we can subtract one unit of time from that. Therefore the best possible scheduled time is $\tau^*_{\text{best}} = |T| - |M|$ since M is maximum. If we consider all schedulings, $\tau^* \geq |T| - |M|$.

Exercise 4.

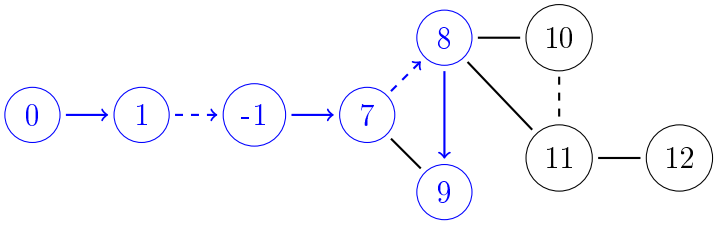
When starting DFS from vertex 0, we first find the two edges to vertex 1 and then 2. From there, there are two options how to continue and because vertex 3 has a lower index then vertex 4, we continue in that direction. We continue with the BFS until we are in the following state:



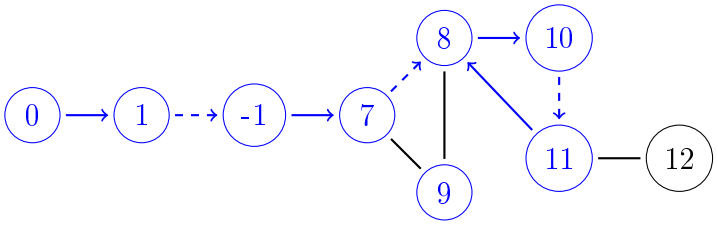
In this state, we found the blossom containing the vertices 2, 3, 4, 5 and 6 and now need to shrink that blossom into a vertex, which we give the index -1 to:



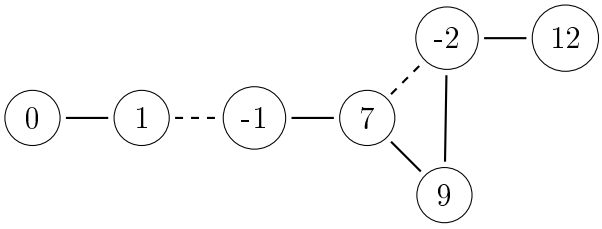
In this modified graph, we start a new DFS from vertex 0 and get to the following state.



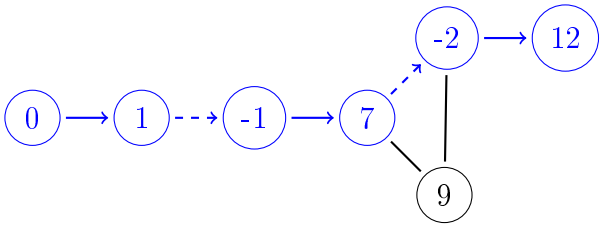
From this state, there is no vertex which we could go to, so we need to backtrack to the last vertex where there is another adjacent edge. In the given state that is vertex 8. If we continue from vertex 8 to 10, next we end up in the following state:



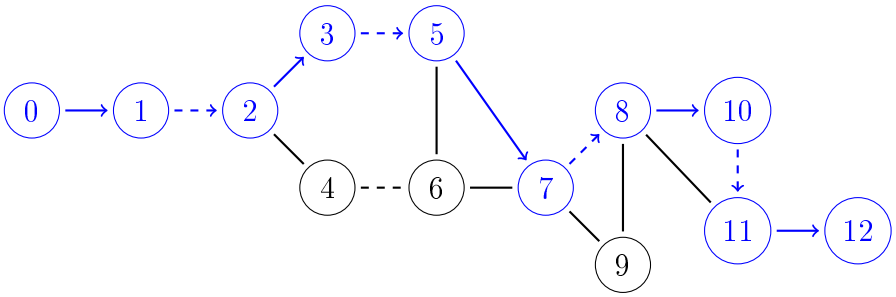
Now we found a blossom containing vertices 8, 10 and 11, which we now shrink:



Now we restart the DFS again and, without the need of backtracking, we find the following augmenting path:



This is the augmenting path which we wanted to find, but before we can augment the matching, we first need to unfold the shrunken blossoms again:



And finally, the matching can be augmented: