Assignment problem

The **assignment problem** is a fundamental combinatorial optimization problem. In its most general form, the problem is as follows:

> The problem instance has a number of *agents* and a number of *tasks*. Any agent can be assigned to perform any task, incurring some *cost* that may vary depending on the agent-task assignment. It is required to perform as many tasks as possible by assigning at most one agent to each task and at most one task to each agent, in such a way that the *total cost* of the assignment is minimized.

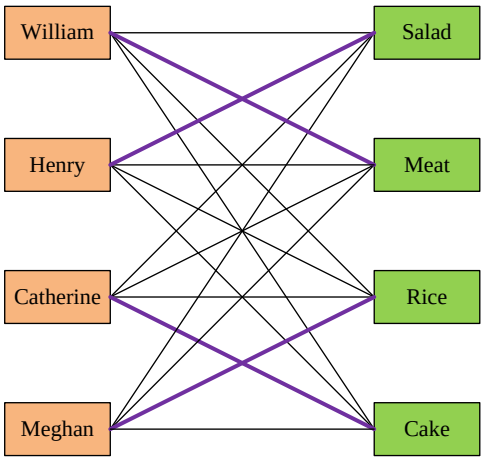
Example. Four friends are going to have a party, and everybody can cook any dish. The table represents the costs of the dishes. The problem is to assign everybody to exactly one dish and to minimize the total cost. We choose exactly one element in every row and in every column.

		Tasks			
		Salad	Meat	Rice	Cake
Agents	William	10	45	8	17
	Henry	9	60	11	20
	Catherine	15	57	14	21
	Meghan	21	48	12	15

Total cost: 9+45+12+21=87.

Alternatively, describing the problem using **graph theory**:

> The assignment problem consists of finding, in a weighted bipartite graph, a matching of a given size, in which the sum of weights of the edges is a minimum.



Linear programming problem:

Variables:

$$x_{i,j} = \begin{cases} 1, & \text{if agent } i \text{ performs task } j, \\ 0, & \text{otherwize.} \end{cases}$$

Objective function:

$$L(X) = \sum_{i,j=1}^{n} c_{i,j} x_{i,j} \to \min_{X}.$$

Constraints:

a) Every agent is assigned to exactly one task.

$$\sum_{j=1}^{n} x_{i,j} = 1.$$

b) Every task is assigned to exactly one agent.

$$\sum_{j=1}^{n} x_{i,j} = 1.$$

c) Non-negative variables.

$$x_{i,j} \geq 0.$$

Dual LP problem.

New variables correspond to the constraints:

Variables for agents:

$$a_j \leftrightarrow \sum_{j=1}^{n} x_{i,j} = 1.$$

Variables for tasks:

$$t_i \leftrightarrow \sum_{i=1}^{n} x_{i,j} = 1.$$

Objective function:

$$l(A,T) = \sum_{j=1}^{n} a_j + \sum_{i=1}^{n} t_i \to \max_{A,T}.$$

Constraints:

$$x_{i,j} \leftrightarrow a_j + t_i \leq c_{i,j}.$$

Primal problem	Dual problem
$L(X) = \sum_{i,j=1}^{n} c_{i,j}x_{i,j} \to \min_X.$ $a_j \leftrightarrow \sum_{j=1}^{n} x_{i,j} = 1$ $t_i \leftrightarrow \sum_{i=1}^{n} x_{i,j} = 1$ $x_{i,j} \geq 0.$	$l(A,T) = \sum_{j=1}^{n} a_j + \sum_{i=1}^{n} t_i \to \max_{A,T}.$ $x_{i,j} \leftrightarrow a_j + t_i \leq c_{i,j}.$

Complementary slackness conditions

$$x_{i,j}(a_j + t_i - c_{i,j}) = 0.$$

So, if $x_{i,j} > 0$, then $a_j + t_i = c_{i,j}$. For all edges (i,j) in the matching, $a_j + t_i = c_{i,j}$. Any matching satisfying these condition minimizes the primal objective function and maximizes the dual objective function.

Hungarian algorithm.

Step 1. Reduce the matrix: find the minimum cost in every row and subtract it from all elements of the row. Then do the same with all columns. The obtained minimums are the values of the dual problem variables (a_j, t_i).

8	10	45	8	17
9	9	60	11	20
14	15	57	14	21
12	21	48	12	15

	0	36	0	3
8	2	37	0	9
9	0	51	2	11
14	1	43	0	7
12	9	36	0	3

	0	36	0	3
8	2	1	0	6
9	0	15	2	8
14	1	7	0	4
12	9	0	0	0

Zeroes in the matrix correspond to constraints where $a_j + t_i - c_{i,j} = 0$. So, we can include the corresponding edges to the matching. In fact, now we have in the cells the *reduced costs*:

$$c'_{i,j} = c_{i,j} - a_i - t_j.$$

Step 2. Construct any initial matching. Color cells with zeroes, having not more than one colored cell in every row and in every column. Further, we don't need values of the variables, but we'll use numbers of rows and columns.

	1	2	3	4
1	2	1	0	6
2	0	15	2	8
3	1	7	0	4
4	9	0	0	0

Step 3. If all rows and columns contain a colored zero, these zeroes provide a solution, go to the end. Otherwise, label a row without colored zeroes by 0.

	1	2	3	4
1	2	1	0	6
2	0	15	2	8
3(0)	1	7	0	4
4	9	0	0	0

Repeat the following actions.

a) Label every *unlabeled* column containing an *uncolored* zero in *labeled* row i by i.

	1	2	3(3)	4
1	2	1	0	6
2	0	15	2	8
3(0)	1	7	0	4
4	9	0	0	0

b) Label every *unlabeled* row containing a *colored* zero in *labeled* column j by j.

	1	2	3(3)	4
1(3)	2	1	0	6
2	0	15	2	8
3(0)	1	7	0	4
4	9	0	0	0

If we label a column without colored zeros, goto Step 5. If there are no uncolored zeroes in labeled rows and unlabeled columns, goto Step 4.

Step 4. Consider the intersection of all *labeled* rows and *unlabeled* columns. Find the minimum value d in these cells.

	1	2	3(3)	4
1(3)	2	**1**	0	6
2	0	15	2	8
3(0)	**1**	7	0	4
4	9	0	0	0

Subtract d to all elements of the *labeled* rows and *unlabeled* columns. Add d to all elements of the *unlabeled* rows and *labeled* columns

	1	2	3(3)	4
1(3)	2	**1**	0	6
2	0	15	2	8
3(0)	**1**	7	0	4
4	9	0	0	0

Here $d = 1$.

	1	2	3(3)	4
1(3)	2−1	**1**−1	0	6−1
2	0	15	2+1	8
3(0)	**1**−1	7−1	0	4−1
4	9	0	0+1	0

Result:

	1	2	3(3)	4
1(3)	1	**0**	0	5
2	0	15	3	8
3(0)	**0**	6	0	3
4	9	0	1	0

Goto Step 3.
Step 3.

	1	2(1)	3(3)	4
1(3)	1	0	0	5
2	0	15	3	8
3(0)	0	6	0	3
4	9	0	1	0

We labeled column 2 without colored zeroes, goto Step 5.

Step 5. Construct an *augmenting path*: start with the column without uncolored zeroes, say j, with label i, include cell (i, j) into the path. Then consider row i, it's labeled by k. If $k = 0$, stop. Otherwise, include cell (i, k) into the path and repeat the Step for column k.

	1	2(1)	3(3)	4
1(3)	1	0	0	5
2	0	15	3	8
3(0)	0	6	0	3
4	9	0	1	0

Augmenting path: $(1,2) \to (1,3) \to (3,3)$ (in violet).

Step 6. Along the augmenting path, change the colors of zeroes: colored zeroes become uncolored, and vice versa, uncolored zeroes become colored.

	1	2	3	4
1	1	0	0	5
2	0	15	3	8
3	0	6	0	3
4	9	0	1	0

Goto Step 3.

Step 3. All rows and columns contain a colored zero, these zeroes provide a solution, go to the end.

		Tasks			
		Salad	Meat	Rice	Cake
Agents	William	10	45	8	17
	Henry	9	60	11	20
	Catherine	15	57	14	21
	Meghan	21	48	12	15

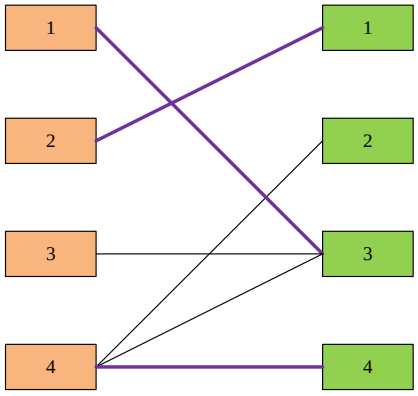
They have a happy party! The total cost is 9+45+14+15=83.

Explanation of the algorithm.

Step 3.

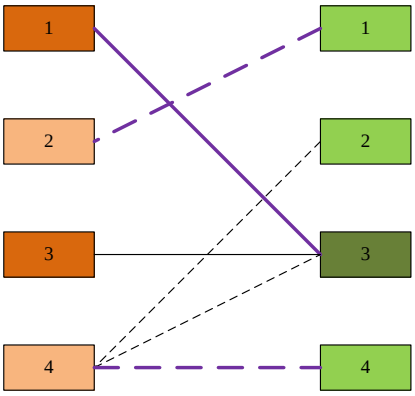
A row (a column) without colored zeroes correspond to an unsaturated vertex of the matching. Zeroes correspond to edges that can be included into the matching (these edges are included in *special spanning subgraph*). Colored zeroes correspond to edges from the matching (thick lines).

	1	2	3(3)	4
1(3)	2	1	0	6
2	0	15	2	8
3(0)	1	7	0	4
4	9	0	0	0



Labeling rows and columns, we construct an *alternating tree* − a tree in which the edges in every path starting in the root and ending in a leave are alternately *out of* and *in* the matching (*uncolored* and *colored* zeroes). In the picture, it is indicated by solid line. The root is vertex 3 in the left part.

	1	2	3(3)	4
1(3)	2	1	0	6
2	0	15	2	8
3(0)	1	7	0	4
4	9	0	0	0

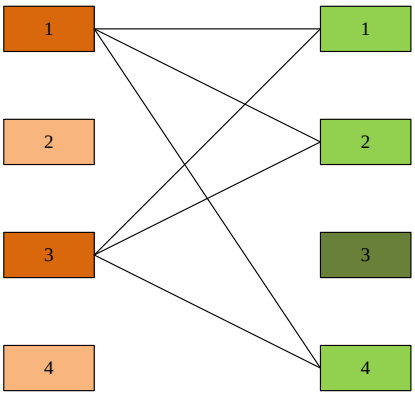


If there are no labeled rows with uncolored zeroes then we can't proceed constructing the tree: in our subgraph, there are no other edges out of the matching, incident with marked vertices in the left part (3 and 1 orange, in our case). So, we update the values of the variables and modify the subgraph. All edges in matching should stay in the subgraph.

Step 4. Consider the intersection of all *labeled* rows and *unlabeled* columns. Find the minimum value *d* in these cells.

	1	2	3(3)	4
1(3)	2	**1**	0	6
2	0	15	2	8
3(0)	**1**	7	0	4
4	9	0	0	0

In fact, we consider all edges, starting in marked orange vertices and ending in unmarked green vertices.



To proceed with the tree, we need to include one of these edges to our special spanning subgraph. For this purpose, we recalculate the values of the variables, and hence, the *reduced costs*:

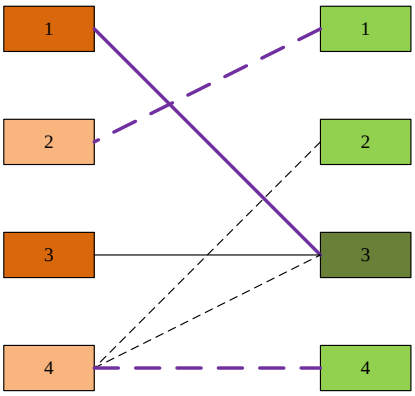
$$c'_{i,j} = c_{i,j} - a_i - t_j.$$

We need to turn into zero at least one cost $c'_{i,j}$ where i correspond to a marked row (agent) and j correspond to an unmarked row (task). As $c'_{i,j}$ should be non-negative, we find the minimum value of $c'_{i,j}$, say d. Then, we update the variables and recalculate $c'_{i,j}$.

$$\tilde{c}'_{i,j} = c_{i,j} - \tilde{a}_i - \tilde{t}_j.$$

	Marked columns	Unmarked columns
Marked rows	$\tilde{a}_i = a_i + d,$ $\tilde{t}_j = t_j - d$ $\tilde{c}'_{i,j} = c_{i,j} - (a_i + d) - (t_j - d);$ $\tilde{c}'_{i,j} = c'_{i,j}$	$\tilde{a}_i = a_i + d,$ $\tilde{t}_j = t_j$ $\tilde{c}'_{i,j} = c_{i,j} - (a_i + d) - t_j;$ $\tilde{c}'_{i,j} = c'_{i,j} - d$
Unmarked rows	$\tilde{a}_i = a_i,$ $\tilde{t}_j = t_j - d$ $\tilde{c}'_{i,j} = c_{i,j} - a_i - (t_j - d);$ $\tilde{c}'_{i,j} = c'_{i,j} + d$	$\tilde{a}_i = a_i,$ $\tilde{t}_j = t_j$ $\tilde{c}'_{i,j} = c_{i,j} - a_i - t_j;$ $\tilde{c}'_{i,j} = c'_{i,j}$

Note that for all edges in matching, the ends are either both marked or both unmarked (according the procedure of the tree constructing).



As you can see, in both these cases the costs don't change; so, the reduced costs of all edges in the matching stay zeroes. The same is true about edges from the tree; both their ends are marked. As for edges with marked orange end and unmarked green end, their costs stay non-negative, because d is the minimum cost among these edges; but some of them become zeroes. It allows us to proceed the tree constructing.

Show the process by using matrices.

	1	2	3(3)	4
1(3)	2	**1**	0	6
2	0	15	2	8
3(0)	**1**	7	0	4
4	9	0	0	0

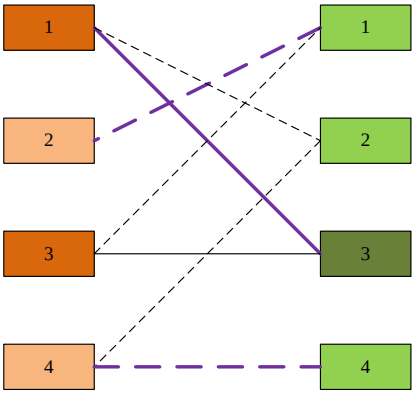
Here $d = 1$.

	1	2	3(3)	4
1(3)	2−1	**1**−1	0	6−1
2	0	15	2+1	8
3(0)	**1**−1	7−1	0	4−1
4	9	0	0+1	0

Result:

	1	2	3(3)	4
1(3)	1	**0**	0	5
2	0	15	3	8
3(0)	**0**	6	0	3
4	9	0	1	0

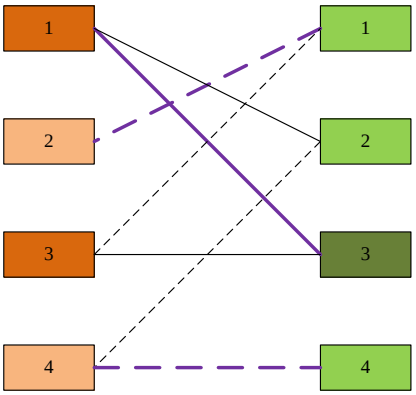
At the graph:



After that, we goto Step 5 and proceed the tree constructing.

	1	2(1)	3(3)	4
1(3)	1	0	0	5
2	0	15	3	8
3(0)	0	6	0	3
4	9	0	1	0

We labeled column 2 without colored zeroes, it means, we came to green vertex 2 which is not covered by the matching. Goto Step 5.



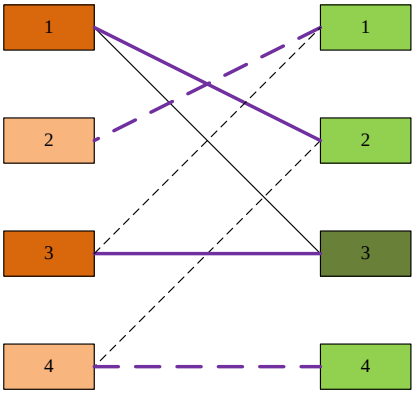
Step 5. We have an augmented path – a path where the edges are alternately *out of* and *in* the matching, and the ends are not covered by the matching (in the table, we have uncolored and colored zeroes, in violet).

	1	2(1)	3(3)	4
1(3)	1	0	0	5
2	0	15	3	8
3(0)	0	6	0	3
4	9	0	1	0

Step 6. Along the augmenting path, change the colors of zeroes: colored zeroes become uncolored, and vice versa, uncolored zeroes become colored. As the number of uncolored zeroes exceed the number of colored ones, we increase the number of edges in the matching.

	1	2	3	4
1	1	0	0	5
2	0	15	3	8
3	0	6	0	3
4	9	0	1	0

At the graph:



We found a perfect matching satisfying the complementary slackness conditions:

$$x_{i,j}\left(a_j + t_i - c_{i,j}\right) = 0.$$

For all $x_{i,j} = 1$, $\left(a_j + t_i - c_{i,j}\right)$=0.