ЗАДАЧА ЗА НАЗНАЧЕНИЯТА. УНГАРСКИ АЛГОРИТЪМ

6	7	3	16	12
11	20	14	21	30
1	9	3	16	13
8	4	17	28	13
13	14	5	19	26

Задача за назначенията:

Даден е тегловен двуделен граф с по n върха във всеки дял.
Търси се сдвояване (съчетание) от n ребра с най-голям сбор на теглата.

В матрицата на графа, съставена от теглата на неговите ребра, търсим n елемента, по един от ред и стълб, с възможно най-голям сбор.

– 6	– 7	– 3	– 16	– 12
– 11	– 20	– 14	– 21	– 30
– 1	– 9	– 3	– 16	– 13
– 8	– 4	– 17	– 28	– 13
– 13	– 14	– 5	– 19	– 26

Тълкуване: Имаме n работници и n работи. Знаем с каква ефективност всеки работник върши всяка работа. Търсим назначение на работниците, при което всеки работник да бъде назначен само на едно работно място, всяка работа да се върши само от един работник, а общата ефективност да бъде възможно най-голяма.

Унгарски алгоритъм:

Стъпка 1: Сменяме знаците на всички тегла.
Оттук нататък вече търсим най-малък вместо най-голям сбор.

7	13	14	12	18
2	0	3	7	0
12	11	14	12	17
5	16	0	0	17
0	6	12	9	4

Стъпка 2: Числата във всеки стълб събираме с най-голямото число в същия стълб на първоначалната матрица.

Прибавянето на едно и също число x към всички числа в даден стълб увеличава всички сборове с x, включително търсения най-малък сбор, но не променя назначението, при което се достига той.

След тази стъпка всички числа в матрицата стават неотрицателни. На следващите стъпки числата в матрицата остават неотрицателни, поради което алгоритъмът се стреми да получи нулев сбор. За целта алгоритъмът се опитва да избере n нули, по една от ред и стълб. Стъпка 2 осигурява поне една нула във всеки стълб на матрицата, но не непременно във всеки ред.

0	6	7	5	11
2	0	3	7	0
1	0	3	1	6
5	16	0	0	17
0	6	12	9	4

Стъпка 3: Най-малкото число във всеки ред на получената матрица изваждаме от всички числа в същия ред.

След тази стъпка всички числа в матрицата остават неотрицателни, но сега вече има нула във всеки ред (и във всеки стълб) на матрицата. Оптималното назначение е останало същото, въпреки че стойността му може да се е променила.

0 *	6	7	5	11
2	0 *	3	7	0
1	0	3	1	6
5	16	0 *	0	17
0	6	12	9	4

Стъпка 4: Отбелязваме със звездички възможно най-много нули, по една от ред и стълб. Ако сме отбелязали n нули, то техните места задават оптимално назначение и алгоритъмът приключва работа.

Постигането на действителния най-голям брой нули е трудна задача, затова прилагаме алчен алгоритъм: отбелязваме всяка срещната нула, в чийто ред и стълб все още няма нула, отбелязана със звездичка. Така може да попаднем на локален максимум, който не е абсолютен.

0 *	6	7	5	11
2	0 *	3	7	0
1	0	3	1	6
5	16	0 *	0	17
0	6	12	9	4
+	+	+		

В случай че нулите, отбелязани със звездички, са по-малко от n, алгоритъмът продължава работа.

Стъпка 5: Отбелязваме с плюсове стълбовете, в които има 0 *.

Наричаме заети онези редове и стълбове, които са отбелязани с плюс.
Наричаме заети онези числа от матрицата, чийто ред или стълб е зает.
Едно число е незаето само ако нито редът му, нито стълбът му е зает.

Стъпка 6: Ако най-малкото незаето число е положително, вадим го от числата в незаетите редове и го събираме със заетите стълбове.

След тази стъпка всички числа в матрицата остават неотрицателни и оптималното назначение не е променено, но се появява незаета нула.

0 *	6	7	5	11
2	0 *	3	7	0
1	0	3	1	6
5	16	0 *	0 ′	17
0	6	12	9	4
+	+	+		

Стъпка 7: Отбелязваме произволна незаета нула със знака прим. Ако в нейния ред има 0 *, преминаваме към стъпка 8 от алгоритъма. В противен случай отиваме на стъпка 9.

0 *	6	7	5	11	
2	0 *	3	7	0	
1	0	3	1	6	
5	16	0 *	0 ′	17	+
0	6	12	9	4	
+	+				

Стъпка 8: Изтриваме знака плюс от стълба, в който е тази 0 *. Отбелязваме с плюс реда, в който се намира току-що означената 0 ′. Отиваме на стъпка 6.

Идеята на тази стъпка е, че нулата, която току-що отбелязахме с прим, е била избрана лошо: тя не може да бъде отбелязана със звездичка, тъй като в нейния ред вече има 0 *. Като преместваме знака плюс, обявяваме тази 0 ′ за заета (и оставяме заета съответната 0 *), затова новото изпълнение на стъпки 6 и 7 ще търси друга незаета нула.

0 *	6	7	5	11	
2	0 *	3	7	0 ′	+
1	0	3	1	6	
5	16	0 *	0 ′	17	+
0	6	12	9	4	
+					

В конкретния пример следващата незаета нула, отбелязана с прим, също има 0 * в реда си, поради което преместваме още един плюс.

Стъпки 6, 7 и 8 образуват цикъл, чрез който търсим незаета нула, в чийто ред няма 0 *.

0 *	6	7	5	11	
2	0 *	3	7	0 ′	+
1	0 ′	3	1	6	
5	16	0 *	0 ′	17	+
0	6	12	9	4	
+					

Стъпка 9: Започвайки от намерената 0 ′, в чийто ред липсва 0 *, описваме верига от белязани нули — от 0 ′ по нейния стълб до 0 *, от тази 0 * по нейния ред до 0 ′ и тъй нататък, докато е възможно. Премахваме звездичките от нулите във веригата, после заменяме в нея всички знаци прим със звездички. Изтриваме всички други знаци прим и всички плюсове.
Ако има n нули, отбелязани със звездички, то техните места образуват оптимално назначение, затова алгоритъмът прекратява изпълнението си. В противен случай се изпълнява стъпка 5.

0 *	6	7	5	11
2	0	3	7	0 *
1	0 *	3	1	6
5	16	0 *	0	17
0	6	12	9	4
+	+	+		+

Стъпка 9 построява алтернираща път и така увеличава с една нулите, отбелязани със звездички. Алгоритъмът намира решение, т.е. отбелязва n клетки със звездички, след най-много $n – 1$ изпълнения на стъпка 9.

Унгарският алгоритъм е предложен от Егервари. Първоначалният вариант има сложност $\Theta\left(n^4\right)$. Съществуват оптимизации със сложност $\Theta\left(n^3\right)$.

Когато задачата има няколко оптимални назначения, алгоритъмът намира едно от тях, а не всичките.

0 *	6	7	5	11
2	0	3	7	0 *
1	0 *	3	1	6
5	16	0 *	0 ′	17
0	6	12	9	4
+	+	+		+

Приложеният числов пример е взет от статията "Задача за назначенията" с автор Димитър Иванчев от София, публикувана в сп. "Математика", бр. 9–10 от 1990 г.

0 *	6	7	5	11	
2	0	3	7	0 *	
1	0 *	3	1	6	
5	16	0 *	0 ′	17	+
0	6	12	9	4	
+	+			+	

0 *	6	6	4	11	
2	0	2	6	0 *	
1	0 *	2	0 ′	6	
6	17	0 *	0 ′	18	+
0	6	11	8	4	
+	+			+	

0 *	6	6	4	11	
2	0	2	6	0 *	
1	0 *	2	0 ′	6	+
6	17	0 *	0 ′	18	+
0	6	11	8	4	
+				+	

0 *	6	6	4	11	
2	0 ′	2	6	0 *	
1	0 *	2	0 ′	6	+
6	17	0 *	0 ′	18	+
0	6	11	8	4	
+				+	

0 *	6	6	4	11	
2	0 ′	2	6	0 *	+
1	0 *	2	0 ′	6	+
6	17	0 *	0 ′	18	+
0	6	11	8	4	
+					

0 *	6	6	4	11	
2	0 ′	2	6	0 *	+
1	0 *	2	0 ′	6	+
6	17	0 *	0 ′	18	+
0	6	11	8	4	
+					

0 *	2	2	0	7	
6	0 ′	2	6	0 *	+
5	0 *	2	0 ′	6	+
10	17	0 *	0 ′	18	+
0	2	7	4	0	
+					

0 *	2	2	0	7	
6	0 ′	2	6	0 *	+
5	0 *	2	0 ′	6	+
10	17	0 *	0 ′	18	+
0	2	7	4	0 ′	
+					

0 *	2	2	0	7	
6	0 ′	2	6	0 *	+
5	0 *	2	0 ′	6	+
10	17	0 *	0 ′	18	+
0	2	7	4	0 ′	
+					

0 *	2	2	0	7
6	0 *	2	6	0
5	0	2	0 *	6
10	17	0 *	0	18
0	2	7	4	0 *

0 *	2	2	0	7
6	0 *	2	6	0
5	0	2	0 *	6
10	17	0 *	0	18
0	2	7	4	0 *

6	7	3	16	12
11	20	14	21	30
1	9	3	16	13
8	4	17	28	13
13	14	5	19	26

В този конкретен пример оптималното назначение постига ефективност 85.

На една предишна стъпка имаше две незаети нули, от които избрахме едната. С другата нула намираме друго решение, но отново с ефективност 85.

0 *	2	2	0 ′	7	
6	0 ′	2	6	0 *	+
5	0 *	2	0 ′	6	+
10	17	0 *	0 ′	18	+
0	2	7	4	0	
+					

0 *	2	2	0 ′	7	+
6	0 ′	2	6	0 *	+
5	0 *	2	0 ′	6	+
10	17	0 *	0 ′	18	+
0	2	7	4	0	

0 *	2	2	0 ′	7	+
6	0 ′	2	6	0 *	+
5	0 *	2	0 ′	6	+
10	17	0 *	0 ′	18	+
0	2	7	4	0	

0 *	2	2	0 ′	7	+
6	0 ′	2	6	0 *	+
5	0 *	2	0 ′	6	+
10	17	0 *	0 ′	18	+
0 ′	2	7	4	0	

0	2	2	0 *	7
6	0	2	6	0 *
5	0 *	2	0	6
10	17	0 *	0	18
0 *	2	7	4	0

6	7	3	16	12
11	20	14	21	30
1	9	3	16	13
8	4	17	28	13
13	14	5	19	26

По този начин се получава друго оптимално назначение с обща ефективност 85.