Алгоритм слежения за объектами через нахождение соответствий

Леонид Бейненсон, 2019-07-08

Internet of Things Group

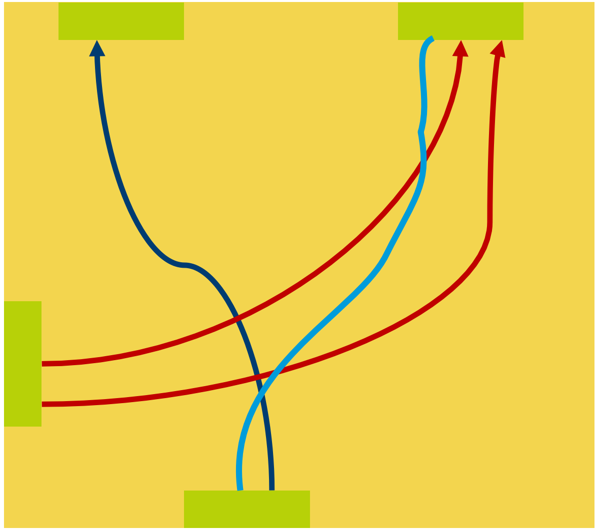
План

- Часть 1. Слежение за пешеходами и задача о назначениях
- Часть 2. Решение задачи о назначениях.

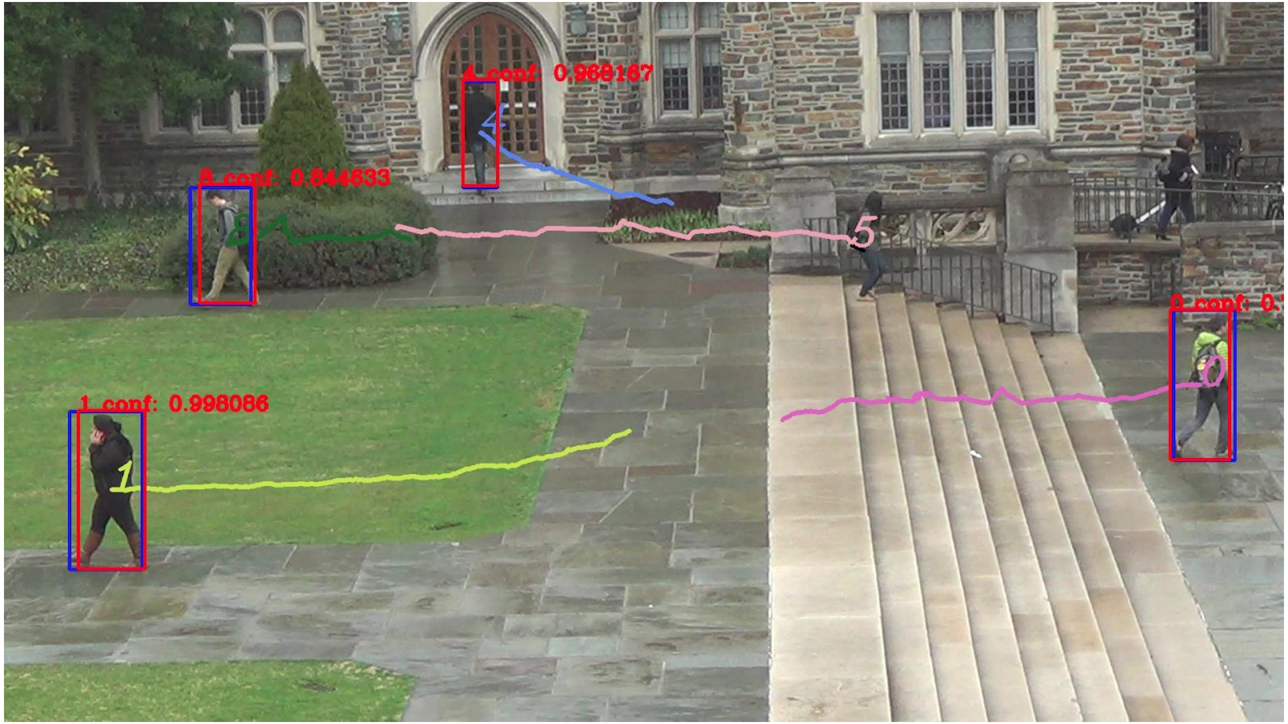
Часть 1. Слежение за пешеходами и задача о назначениях

Пример постановки задачи

- Есть некоторое помещение.
  Несколько входов, несколько выходов.
- По данным с камеры видеонаблюдения нужно определить сколько людей за день проходит из входа i в выход j
- Решение: строить траектории пешеходов



Пример постановки задачи



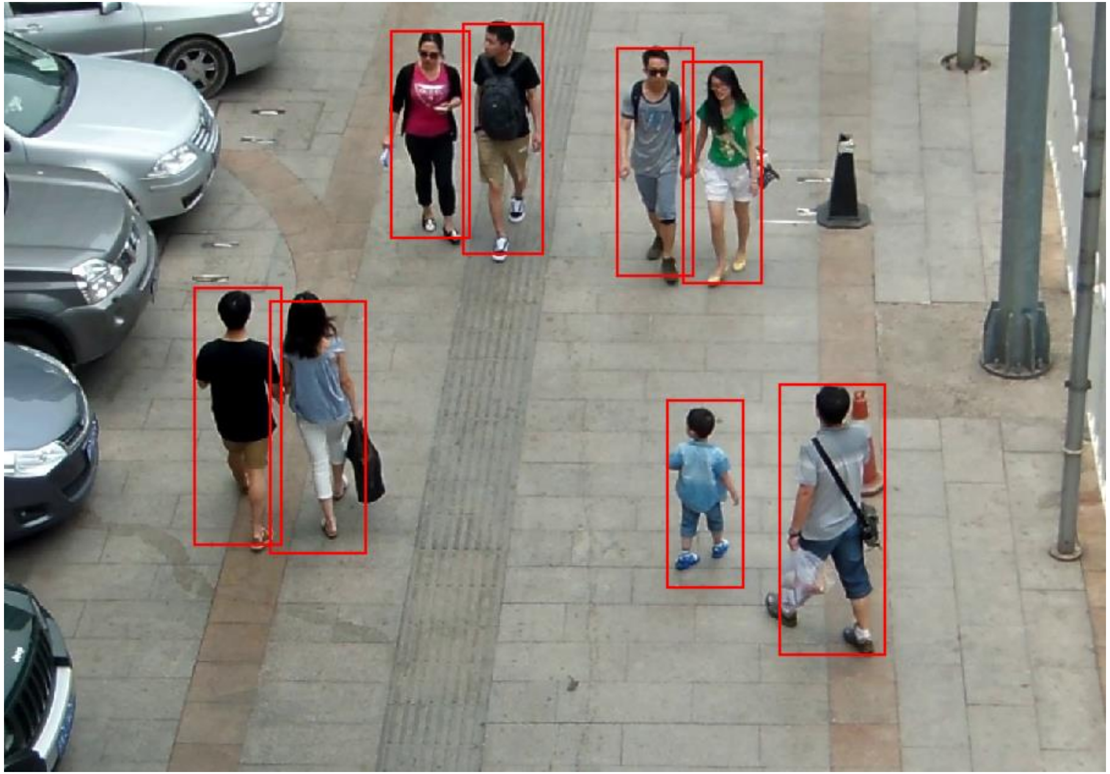
Детектирование пешеходов

- Для детектирования используются нейронные сети.
  Например, MobileNet_SSD, натренированная на датасете PASCAL VOC детектирует оъекты 20 классов:
    - Аэропланы (индекс класса = 1)
    - Велосипеды (индекс класса = 2)

      ...
    - Пешеходы (индекс класса = 15)

      ...
- Сеть возвращает вектор объектов, каждый из которых описывается как
    - Координаты прямоугольника (x_left, x_right, y_top, y_bottom)
    - Индекс класса -- от 1 до 20
    - Confidence (уверенность) – float от 0.0 до 1.0

Детектирование пешеходов



Составление траекторий

Предположим до фрейма № t нашли пешеходов и определили траектории $T_1, \dots, T_N$.

На фрейме № t+1 нашли пешеходов – прямоугольники $R_1, \dots, R_D$.

Как определить траектории?

Решение: вычислить «схожесть» между траекториями и пешеходами: для каждого i от 1 до N, для каждого j от 1 до D определяем

a_{ij} – схожесть (affinity) между траекторией T_i и прямоугольником R_j

$A = (a_{ij})$ – матрица схожести (affinity matrix).

По этой матрице мы сможем определить какой траектории какой задетектированный пешеход принадлежит.

Схема алгоритма

- Получить новый фрейм
- Задетектировать пешеходов на новом фрейме
- Вычислить матрицу схожести A между задетектированными пешеходами и траекториями
- Определить по матрице схожести
    - Каким траекториям какой задетектированный пешеход соответствует
    - Какие пешеходы – новые
(не соответствуют ни одной уже существующей траектории)
    - Какие траектории на этом фрейме – без пешехода
(ушел, зашел за столб, итп)
- Обновить траектории, в частности удалить те из них, которым давно не назначали пешеходов

Схема алгоритма

Как вычисляем матрицу А?

- Получить новый фрейм
- Задетектировать пешеходов на новом фрейме
- Вычислить матрицу схожести А между задетектированными пешеходами и траекториями
- Определить по матрице схожести
  - Каким траекториям какой задетектированный пешеход соответствует
  - Какие пешеходы – новые
    (не соответствуют ни одной уже существующей траектории)
  - Какие траектории на этом фрейме – без пешехода
    (ушел, зашел за столб, итп)
- Обновить траектории, в частности удалит [illegible] давно не назначали пешеходов

Как по матрице А находим соответствия?

Вычисление матрицы схожести A (1/5)

Для вычисления «коэффициента схожести» траектории T_i и нового пешехода R_j самый простой способ:

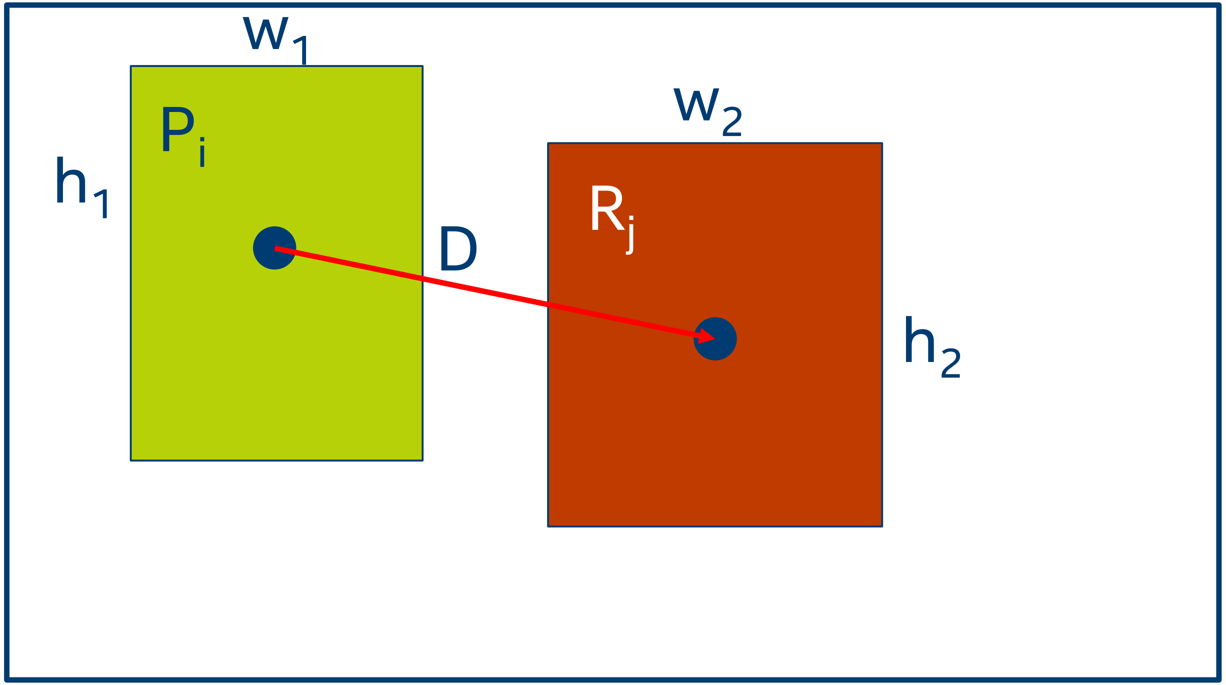
- взять последнее положение траектории T_i в котором мы уверены, – пусть это будет пешеход P_i на фрейме t-k
- и сравнить того пешехода P_i (задетектированного фрейме t-k) и нового пешехода R_j

Как сравнивать:

- По расположению на фрейме
- По размеру
- По внешнему виду

Вычисление матрицы схожести А (2/5)

Сравниваем пешехода P_i и нового пешехода R_j по положению и размеру.



Вычисление матрицы схожести A (3/5)

Сравниваем пешехода P_i и нового пешехода R_j по положению и размеру.

Пусть D – расстояние между центрами прямоугольников P_i и R_j,
(w_1,h_1) – ширина и высота прямоугольника P_i
(w_2, h_2) – ширина и высота прмоугольника R_j
выберем какие-то веса $C_1 > 0$ и $C_2 > 0$

Коэффициент сходства по положению на фрейме:

$$affinity\_place = \exp(-C_1 * D^2 / (w_1 * h_1))$$

Коэффициент сходства по размеру:

$$affinity\_shapes = \exp(-C_2 * (|w_1-w_2|/w_1 + |h_1-h_2|/h_1))$$

Вычисление матрицы схожести A (4/5)

Сравниваем пешехода P_i и нового пешехода R_j по внешнему виду.

Простой способ сравнения:

- Вырезать из фрейма t прямоугольник R_j,
- Вырезать из фрейма t-k прямоугольник P_i,
- Уменьшить размеры вырезанных картинок до 16x32 (т.е. 16 в ширину, 32 в высоту)
- Вычислить affinity_appearance как **кросс-корреляцию между уменьшенными изображениями**

Сложный способ сравнения – натренировать **сетку**, которая по вырезанной картинке выдает вектор (например, float[256]), описывающий внешний вид пешехода так, чтобы **косинус угла между векторами** был метрикой сходства.

Вычисление матрицы схожести A (5/5)

Собираем общую метрику сходства пешехода P_i и нового пешехода R_j как

a_{ij} = affinity_place * affinity_shapes * affinity_appearance

Это дает нам общую метрику сходства между траекторией и новым задетектированным пешеходом.

При этом все affinity-коэффициенты у нас будут $0 \leq a_{ij} \leq 1$.

Нахождение соответствий траектории-пешеходы по матрице схожести A (1/6)

В простом случае можем найти соответствия сразу.

Простой случай – это когда:

- в каждой строке i есть только один максимальный элемент с индексом f(i)
- Для разных строк эти элементы разные

$$\forall\, i \neq j \text{ выполняется } f(i) \neq f(j)$$

– то есть, каждый новый пешеход подходит только одной траектории, а траектории не конкурируют за новых пешеходов

$$A = \begin{pmatrix} 0.1 & 0.9 & 0.0 & 0.2 \\ 0.2 & 0.0 & 0.6 & 0.2 \\ 0.3 & 0.1 & 0.1 & 0.2 \\ 0.6 & 0.7 & 0.8 & 0.9 \end{pmatrix}$$

f(1) = 2

f(2) = 3

f(3) = 1

f(4) = 4

Нахождение соответствий траектории-пешеходы по матрице схожести A (2/6)

Дополнительный шаг после того, как нашли соответствия – **не назначать** трактории T_i нового пешехода R_j, если a_{ij} меньше некоторого порога.

В этом случае будем считать, что траектории **не нашлось соответствий**.

Если некоторому прямоугольнику R_j не нашлось соответствий – начало **новой** траектории

$$A = \begin{pmatrix} 0.1 & 0.9 & 0.0 & 0.2 \\ 0.2 & 0.0 & 0.6 & 0.2 \\ 0.3 & 0.1 & 0.1 & 0.2 \\ 0.6 & 0.7 & 0.8 & 0.9 \end{pmatrix}$$

Если порог равен 0.5:

f(1) = 2

f(2) = 3

f(3) = **None**

f(4) = 4

Нахождение соответствий траектории-пешеходы по матрице схожести A (3/6)

Сложный случай – когда траектории начинают конкурировать за пешеходов.

В частности, такое может быть, когда задетектировано больше пешеходов, чем было траекторий
– нужно выбрать, какие пешеходы будут начинать новые траектории, а какие продолжать старые.

$$A = \begin{pmatrix} 0.1 & 0.9 & 0.0 & 0.2 \\ 0.2 & 0.0 & 0.6 & 0.2 \\ 0.2 & 0.1 & 0.8 & 0.2 \end{pmatrix}$$

Нахождение соответствий траектории-пешеходы по матрице схожести A (4/6)

В сложном случае можем сформулировать задачу так: назначить траекториям $T_1,\dots,T_N$ новых пешеходов R_1 , …, R_D по матрице сходства A так, чтобы

- Одной траектории i соответствовало не более одного нового пешехода f(i), то есть

$$f : \{1, \dots, N\} \to \{1, \dots, D\} \cup \{None\}$$

- Каждый пешеход назначается не более чем одной траектории, то есть

$$\forall i_1 \neq i_2 \quad \text{если } f(i_1) \neq None \text{ и } f(i_2) \neq None, \text{то } f(i_1) \neq f(i_2)$$

- сумма a_{ij} по назначениям была максимальна:

$$\sum_i a_{i,f(i)} \to \max$$

Нахождение соответствий траектории-пешеходы по матрице схожести A (5/6)

В сложном случае задача поиска соответсвий сводится к **задаче о назначениях**.

В канонической формулировке задача о назначениях выглядит так:

- имеется некоторое число *работ* {1,…,N} и **то же самое** число *исполнителей* {1,…,N}
- любой исполнитель i может быть назначен на выполнение любой (но только одной) работы j = f(i), с затратами a(i,j) ≥ 0
- нужно распределить работы так, чтобы выполнить работы с минимальными суммарными затратами

$$\sum_i a(i, f(i)) \rightarrow min$$

Нахождение соответствий траектории-пешеходы по матрице схожести A (5/6)

В сложном случае задача поиска соответсвий сводится к **задаче о назначениях**.

В канонической формулировке задача о назначениях выглядит так:

- имеется некоторое число *работ* {1,…,N} и **то же самое** число *исполнителей* {1,…,N}
- любой исполнитель i может быть назначен на выполнение любой (но только одной) работы j = f(i), с затратами a(i,j) ≥ 0
- Стоимость назначения спределить работы так, чтобы выполнить работы с ьными суммарными затратами

$$\sum_i a(i, f(i)) \rightarrow min$$

Нахождение соответствий траектории-пешеходы по матрице схожести A (6/6)

Чтобы свести нашу задачу к задаче о назначениях нужно

1. Сделать матрицу A квадратной
   Для этого мы можем добавить некоторое количество «фиктивных» строк и столбцов, заполненных нулями

2. Перейти от задачи «найти максимум» к задаче «найти минимум».
   Для этого мы воспользуемся тем, что у нас $0 \le a_{ij} \le 1$.
   Мы вычтем из 1.0 все значения матрицы A:
   $$a'(i,j) = 1 - a_{ij}$$
   В результате у нас получится неотрицательная матрица A' и если мы решим задачу минимизации $\sum_i a'(i, f(i)) \to min$ это будет эквивалентно исходной задаче

Часть 2. Решение задачи о назначениях

Свойства задачи о назначениях

Мы ищем назначение f минимизируя критерий

$$\sum_i a(i, f(i)) \to min$$

Элементарные преобразования:

- вычесть/прибавить из всех ячеек какой-то строки i одно и то же число
- вычесть/прибавить из всех ячеек какого-то столбца j одно и то же число

Если после таких преобразований a(i,j) ≥ 0 , то оптимальное назначение – не изменится

$$\begin{pmatrix} 1 & 4 & 2 \\ 2 & 2 & 1 \\ 1 & 0 & 5 \end{pmatrix} \to \begin{pmatrix} 1 & 4 & 2 \\ 1 & 1 & 0 \\ 1 & 0 & 5 \end{pmatrix} \to \begin{pmatrix} 0 & 4 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{pmatrix}$$

Потенциалы (1/2)

Вместо того, чтобы вычитать значения из строк или столбцов, будем их просто хранить в отдельных векторах:

- pr(i) -- те значения, которые мы могли бы вычитать из строк,
- pc(j) -- те значения, которые мы могли бы вычитать из столбцов.

Тогда, чтобы после вычитания у нас матрица A все равно осталась бы неотрицательной, значения pr(i) и pc(j) должны удовлетворять условию

для всех i и j выполняется $a(i,j) - pr(i) - pc(j) \geq 0$

Такую пару векторов pr(i), pc(j) будем называть *потенциалами*.

Потенциалы (2/2)

Пример

$$A = \begin{pmatrix} 1 & 9 & 0 & 2 \\ 2 & 1 & 6 & 2 \\ 2 & 1 & 8 & 2 \end{pmatrix}$$

$$pr(i) = (0, 1, 1)$$

$$pc(j) = (1, 0, 0, 0)$$

$$a(i,j) - pr(i) - pc(j) = \begin{pmatrix} 0 & 9 & 0 & 2 \\ 0 & 0 & 5 & 1 \\ 0 & 0 & 7 & 1 \end{pmatrix}$$

Стоимость потенциалов

Стоимостью потенциалов pr(i), pc(j) будем называть значение

$$\boldsymbol{C}(pr, pc) = \sum_i pr(i) + \sum_j pc(j)$$

Для любого назначения f и любых потенциалов pr(i), pc(j) будет

$$a\big(i, f(i)\big) \geq pr(i) + pc(f(i))$$

Но при пробегании i от 1 до N значение f(i) также будет пробегать от 1 до N (пусть и не в том порядке), а значит

$$\sum_i a\big(i, f(i)\big) \geq \sum_i pr(i) + \sum_i pc\big(f(i)\big) = \sum_i pr(i) + \sum_j pc(j) = \boldsymbol{C}(pr, pc)$$

Значит, стоимость **любого** назначения ≥ стоимости **любого** потенциала

Стоимость потенциалов

Стоимостью потенциалов pr(i), pc(j) будем называть значение

Стоимость назначения

$$\boldsymbol{C}(pr, pc) = \sum_i pr(i) + \sum_j pc(j)$$

Стоимость потенциала

Для любого назначения f и любых потенциалов pr(i), pc(j) будет

$$a\big(i, f(i)\big) \geq pr(i) + pc(f(i))$$

Но при пробегании i от 1 до N значение f(i) также будет пробегать от 1 до N (пусть и не в том порядке), а значит

$$\sum_i a\big(i, f(i)\big) \geq \sum_i pr(i) + \sum_i pc\big(f(i)\big) = \sum_i pr(i) + \sum_j pc(j) = \boldsymbol{C}(pr, pc)$$

Значит, стоимость **любого** назначения ≥ стоимости **любого** потенциала

Двойственная задача (1/2)

Из этого свойства стоимости вытекает следующее:
Если мы найдем такое назначение f(i) и такие потенциалы pr(i), pc(j), что стоимость этого назначения **совпадает** со стоимостью этих потенциалов, то данное назначение – **оптимальное**.

Ведь ни у какого назначения не может быть стоимость меньше.

Таким образом мы имеем

- прямую задачу: найти оптимальное назначение f(i) с минимальной стоимостью
- двойственную задачу: найти оптимальные потенциалы pr(i),pc(j) с максимальной стоимостью

И можем решать две этих задачи одновременно

Двойственная задача (2/2)

Будем хранить назначение в виде матрицы F(i,j):

- Если f(i) = j, то F(i,j) = 1 (сотруднику i назначена задача j)
- Иначе F(i,j) = 0

Инициализируем матрицу F нулями, и будем изменять одновременно матрицу назначений F(i,j) и потенциалы pr(i), pc(j) так, чтобы
заполняем F(i,j) = 1 только если $a(i,j) - pr(i) - pc(j) = 0$

Если в процессе работы мы получим матрицу F, удовлетворяющую этому условию, такую, что

- в каждой строке i есть ровно один элемент j такой, что F(i,j) = 1
- в каждом столбце j есть ровно один элемент i такой, что F(i,j) = 1

– такая матрица назначений F(i,j) будет оптимальной.

Двойственная задача (2/2)

$$\sum_i a(i, f(i)) = \sum_{(i,j):F(i,j)=1} a(i,j) =$$

$$= \sum_{i,j:F(i,j)=1} (pr(i) + pc(j)) =$$

$$= \sum_i pr(i) + \sum_j pr(j) = \boldsymbol{C}(pr, pc)$$

Будем хранить назначение в виде матрицы

- Если f(i) = j, то F(i,j) = 1 (сотруднику i назн
- Иначе F(i,j) = 0

Инициализируем матрицу F нулями, и будем
матрицу назначений F(i,j) и потенциалы pr(i), pc(j) так, чтобы
заполняем F(i,j) = 1 только если $a(i,j) - pr(i) - pc(j) = 0$

Если в процессе работы мы получим матрицу F, удовлетворяю тому условию, такую, что

- в каждой строке i есть ровно один элемент j такой, что F(i,j)
- в каждом столбце j есть ровно один элемент i такой, что F(i, = 1

– такая матрица назначений F(i,j) будет являться оптимальной.

Матрица кандидатов

Будем в процессе работы хранить матрицу кандидатов C(i,j):

- C(i,j) = 0 – если $pr(i) + pc(j) < a(i,j)$
  это ячейка – “не кандидат”
- C(i,j) = 1 – если $pr(i) + pc(j) = a(i,j)$
  это ячейка – “кандидат” на назначение

Тогда наше условие на матрицу F будет выглядеть так
заполняем F(i,j) = 1 только если C(i,j) = 1

Общая схема алгоритма задачи о назначениях

```
void НАЙТИ_НАЗНАЧЕНИЕ():
	ИНИЦИАЛИЗИРОВАТЬ_ПОТЕНЦИАЛЫ()
	ЗАПОЛНИТЬ_МАТРИЦУ_КАНДИДАТОВ()
	ЖАДНАЯ_ИНИЦИАЛИЗАЦИЯ_F()
	если в каждой строке i есть ячейка F(i,j) == 1:
		выход // нашли решение жадным алгоритмом

	повторять:
	   ЗАПОЛНИТЬ_МАТРИЦУ_КАНДИДАТОВ()
	   ПОПРОБОВАТЬ_ПЕРЕНАЗНАЧИТЬ_БЕЗ_ИЗМЕНЕНИЯ_ПОТЕНЦИАЛОВ()

	   // мы переназначили без изменения потенциалов все, что могли
	   если в каждой строке i есть ячейка F(i,j) == 1:
		   выход

	   // если оказались здесь, значит не смогли назначить работы
	   // всем сотрудникам без изменения потенциалов
	   ИЗМЕНИТЬ_ПОТЕНЦИАЛЫ()
```

Шаг 0: Инициализация

void ИНИЦИАЛИЗИРОВАТЬ_ПОТЕНЦИАЛЫ():

для всех i устанавливаем $pr(i) = \min\{a(i,j), j = 1, \dots, N\}$

для всех j устанавливаем $pc(j) = 0$

// -- очевидно, что при этих значениях выполняется $pr(i) + pc(j) \leq a(i,j)$

```
void ЗАПОЛНИТЬ_МАТРИЦУ_КАНДИДАТОВ():
  для каждого i от 1 до N:
        для каждого j от 1 до N:
            если pr[i] + pc[j] == a(i,j):
                C(i,j) = 1
            иначе:
                C(i,j) = 0
```

Шаг 1 – жадный алгоритм

```
bool ЖАДНАЯ_ИНИЦИАЛИЗАЦИЯ_F():
        для каждого i от 1 до N:
           для каждого кандидата j в строке i такого, что C(i,j) == 1
                 если в строке i матрицы F пока нет значений “1”, то
                       если в столбце j матрицы F пока нет значений “1”, то
                            F(i,j) := 1
                       иначе
                            F(i,j) := 0
```

Общая схема алгоритма задачи о назначениях (rep)

```
void НАЙТИ_НАЗНАЧЕНИЕ():
        ИНИЦИАЛИЗИРОВАТЬ_ПОТЕНЦИАЛЫ()
        ЗАПОЛНИТЬ_МАТРИЦУ_КАНДИДАТОВ()
        ЖАДНАЯ_ИНИЦИАЛИЗАЦИЯ_F()
        если в каждой строке i есть ячейка F(i,j) == 1:
              выход // нашли решение жадным алгоритмом

        повторять:
           ЗАПОЛНИТЬ_МАТРИЦУ_КАНДИДАТОВ()
           ПОПРОБОВАТЬ_ПЕРЕНАЗНАЧИТЬ_БЕЗ_ИЗМЕНЕНИЯ_ПОТЕНЦИАЛОВ()

           // мы переназначили без изменения потенциалов все, что могли
           если в каждой строке i есть ячейка F(i,j) == 1:
                 выход

           // если оказались здесь, значит не смогли назначить работы
           // всем сотрудникам без изменения потенциалов
           ИЗМЕНИТЬ_ПОТЕНЦИАЛЫ()
```

Шаг 2 – переназначение без изменения потенциалов

	j=0	j=1	j=2	j=3	j=4
i=0	**C**		**F,C**		**C**
i=1	**F,C**				
i=2			**C**		
i=3		**C**			**F,C**

Предположим, мы уже назначили некоторые задачи работникам.

Если какой-то работник i остался без задачи, мы можем попытаться найти в его строке ячейки-кандидаты, определить, кем заняты эти работы, и попытаться найти цепочку переназначений, чтобы их освободить.

Шаг 2 – переназначение без изменения потенциалов

	j=0	j=1	j=2	j=3	j=4
i=0	**C**		**F,C**		**C**
i=1	**F,C**				
i=2			**C**		
i=3		**C**			**F,C**

Здесь обозначаем
- буквой **C**, если в ячейке C(i,j)=1
- буквой **F**, если в ячейке F(i,j)=1

… строке ячейки-кандидаты, определить, кем заняты эти работы, и попытаться найти цепочку переназначений, чтобы их освободить.

Шаг 2 – переназначение без изменения потенциалов

	j=0	j=1	j=2	j=3	j=4
i=0	C		F,C		C
i=1	F,C				
i=2			C		
i=3		C			F,C

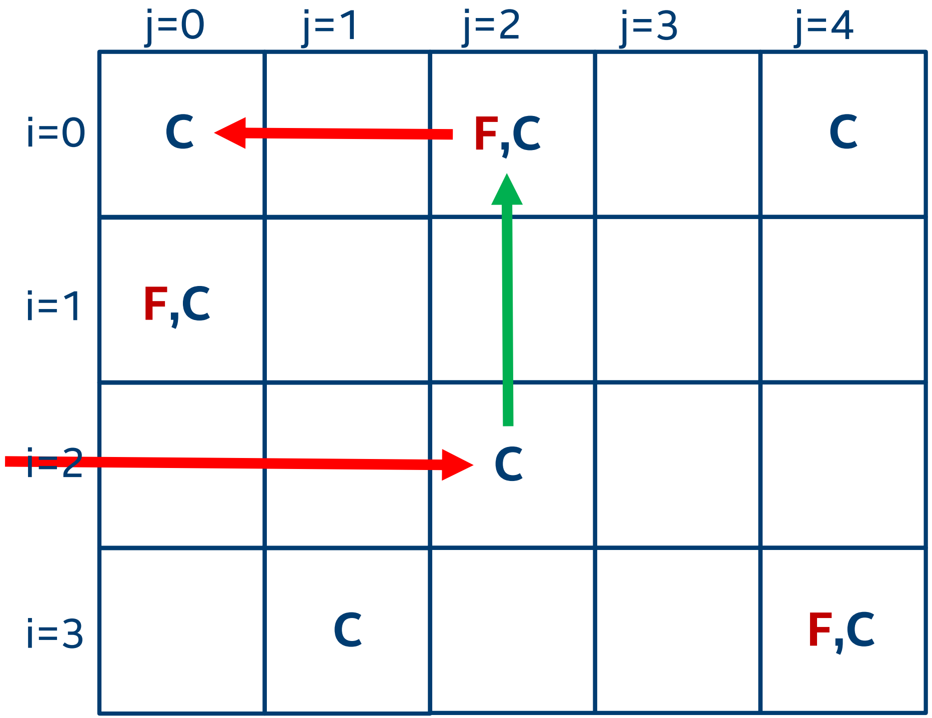
- i = 2 – свободен
  попробовать занять работу j=2

Шаг 2 – переназначение без изменения потенциалов

	j=0	j=1	j=2	j=3	j=4
i=0	C		F,C		C
i=1	F,C				
i=2			C		
i=3		C			F,C

- i = 2 – свободен
  попробовать занять работу j=2
- работа j=2 занята работником i=0

Шаг 2 – переназначение без изменения потенциалов



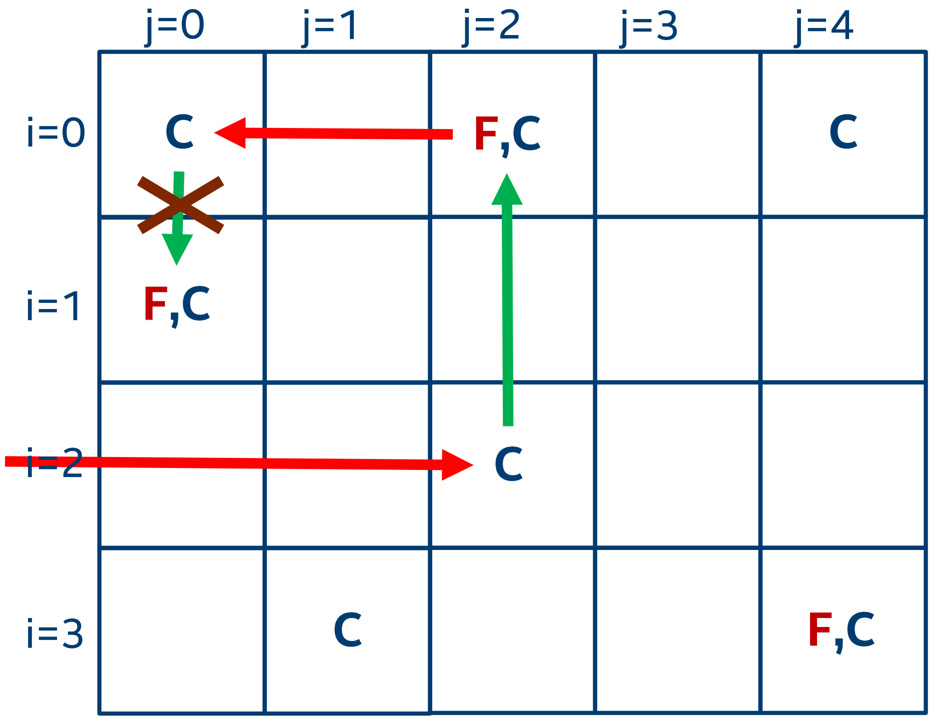
- i = 2 – свободен
  попробовать занять работу j=2
- работа j=2 занята работником i=0
- попытаться переназначить работника i=0 на задачу j=0

Шаг 2 – переназначение без изменения потенциалов

	j=0	j=1	j=2	j=3	j=4
i=0	C		F,C		C
i=1	F,C				
i=2			C		
i=3		C			F,C

- i = 2 – свободен
  попробовать занять работу j=2
- работа j=2 занята работником i=0
- попытаться переназначить работника i=0 на задачу j=0
- задача j=0 занята работником i=1

Шаг 2 – переназначение без изменения потенциалов



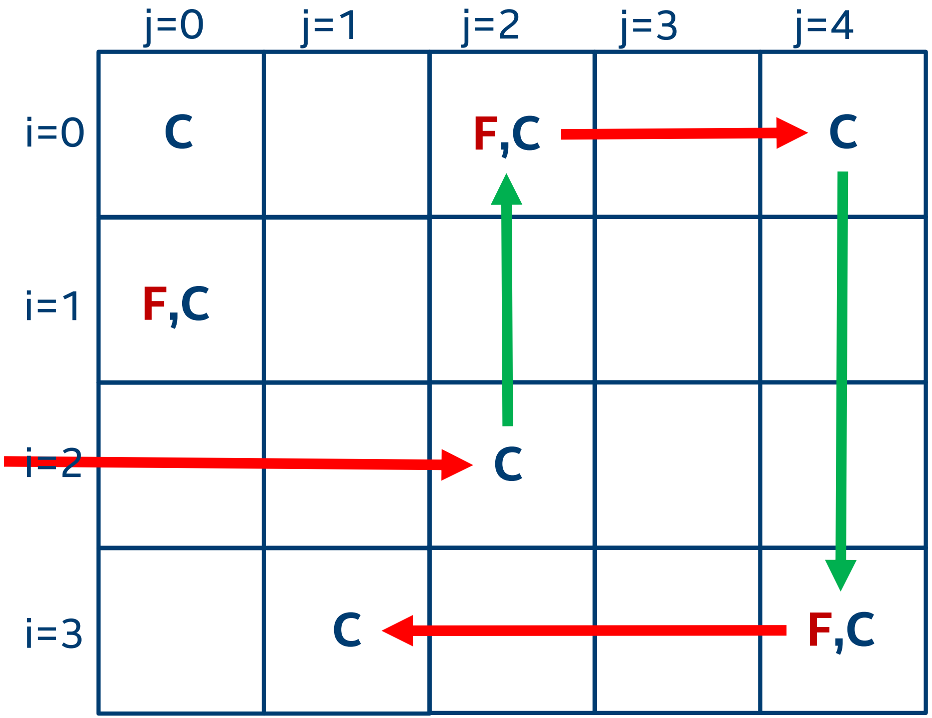
- i = 2 – свободен
  попробовать занять работу j=2
- работа j=2 занята работником i=0
- попытаться переназначить работника i=0 на задачу j=0
- задача j=0 занята работником i=1
- работник i=1 не может взять другую задачу

Шаг 2 – переназначение без изменения потенциалов

	j=0	j=1	j=2	j=3	j=4
i=0	C		F,C		C
i=1	F,C				
i=2			C		
i=3		C			F,C

- i = 2 – свободен
  попробовать занять работу j=2
- работа j=2 занята работником i=0
- ~~попытаться переназначить работника i=0 на задачу j=0~~
- ~~задача j=0 занята работником i=1~~
- ~~работник i=1 не может взять другую задачу~~
- тогда попытаться переназначить работника i=0 на задачу j=4

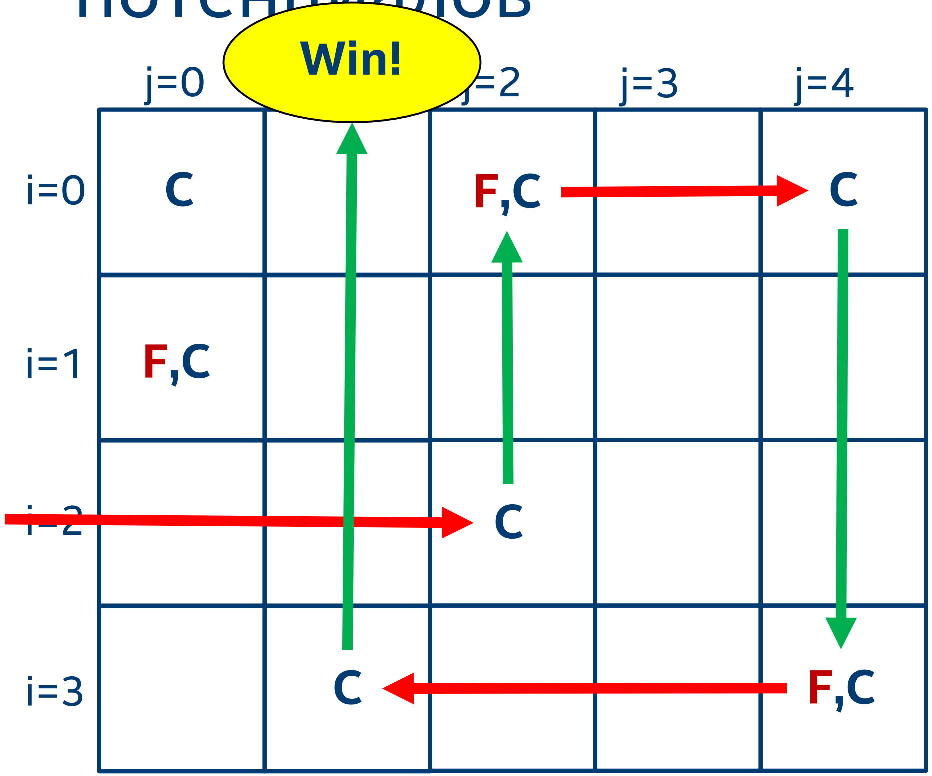
Шаг 2 – переназначение без изменения потенциалов



- i = 2 – свободен
  попробовать занять работу j=2
- работа j=2 занята работником i=0

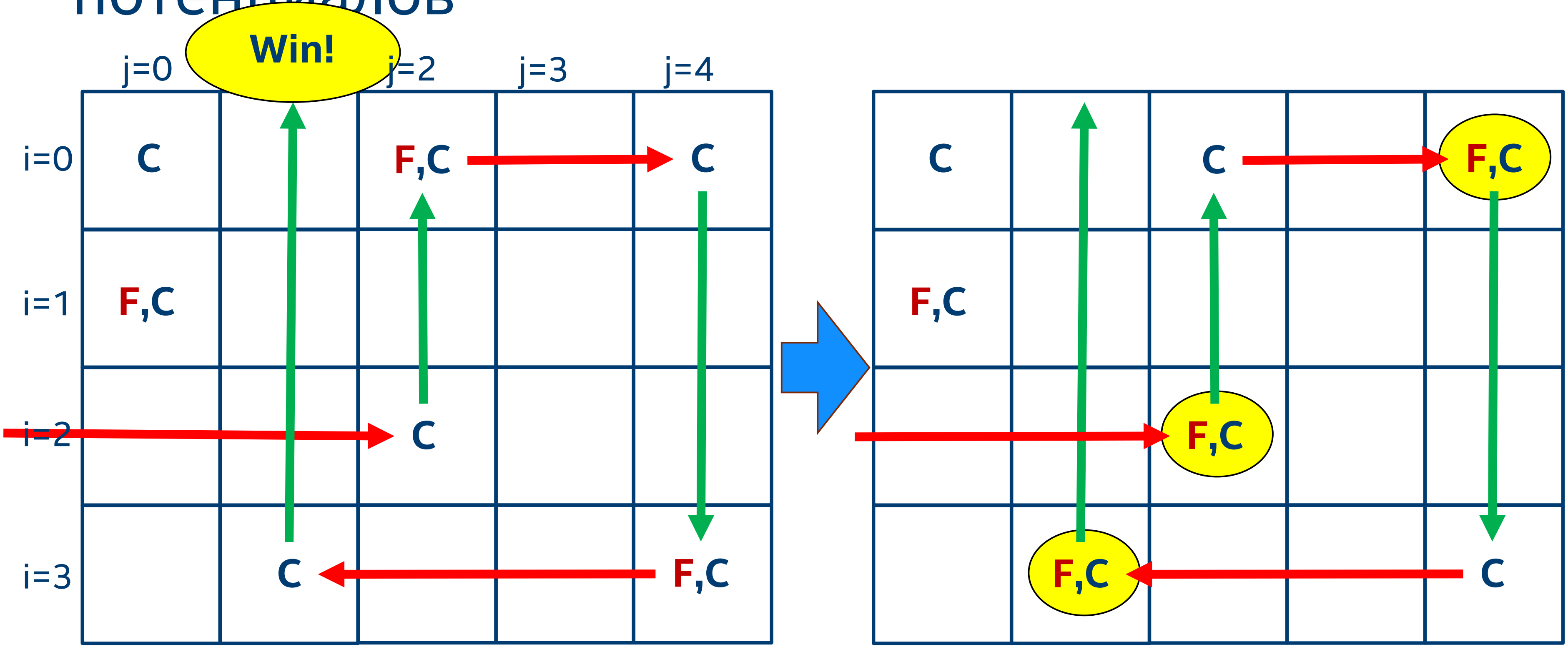
  …
- тогда попытаться переназначить работника i=0 на задачу j=4
- задача j=4 занята работником i=3
- а работника i=3 можно переназначить на задачу j=1

Шаг 2 – переназначение без изменения потенциалов

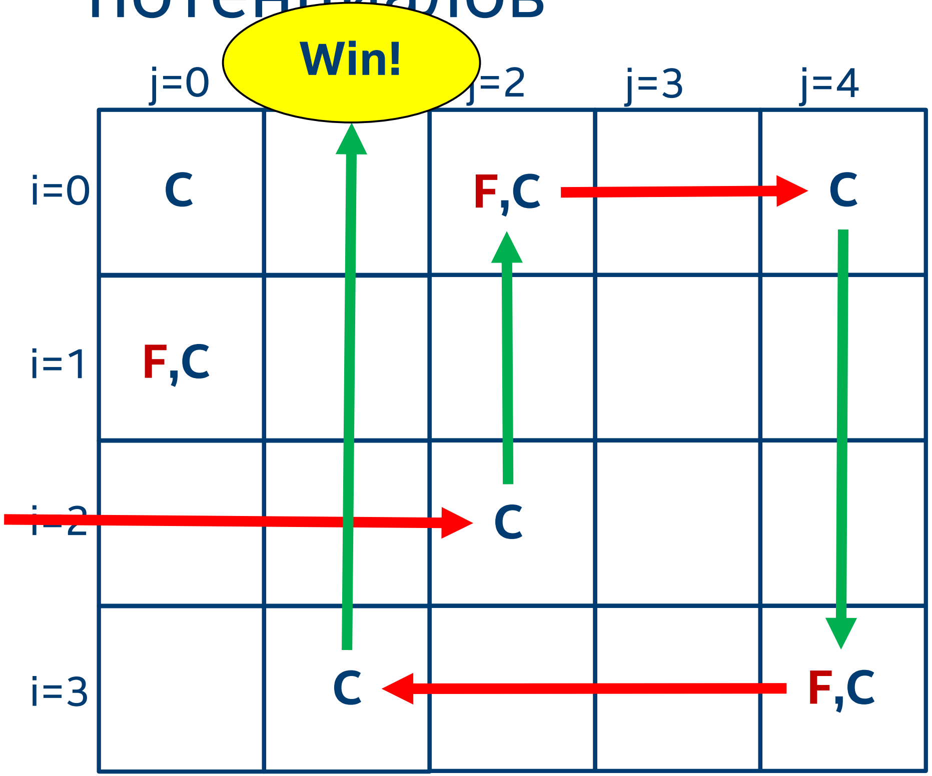


- i = 2 – свободен
  попробовать занять работу j=2
- работа j=2 занята работником i=0
  …
- тогда попытаться переназначить работника i=0 на задачу j=4
- задача j=4 занята работником i=3
- а работника i=3 можно переназначить на задачу j=1
- А задача j=1 – свободна!

Шаг 2 – переназначение без изменения потенциалов



Шаг 2 – переназначение без изменения потенциалов

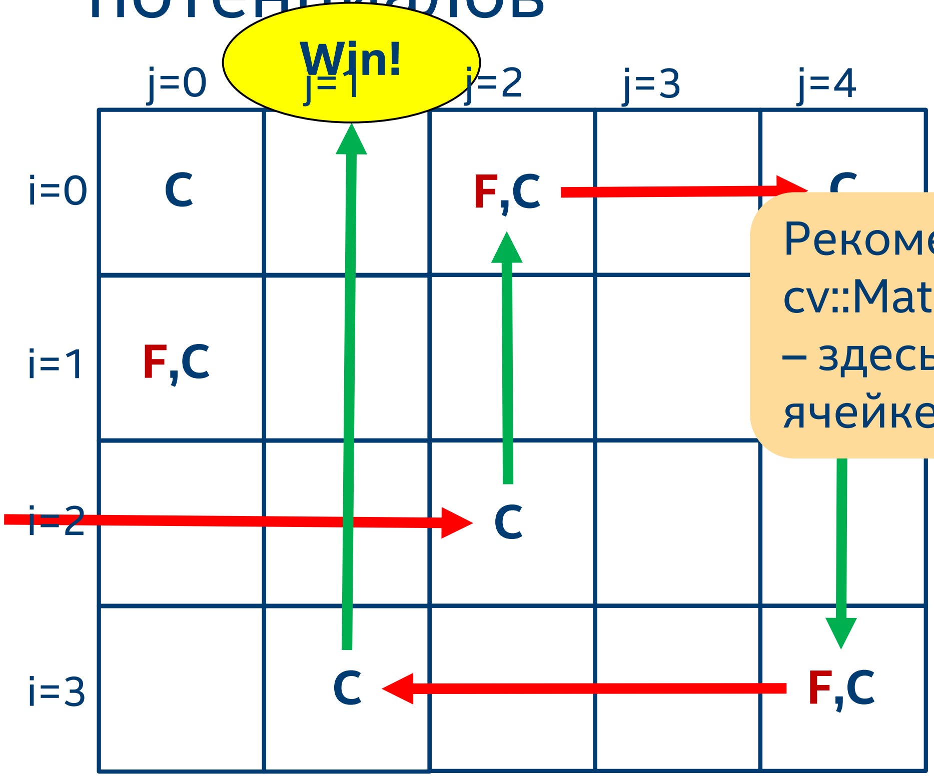


Как эти метания оформить алгоритмически?

Будем хранить

- вектор zr(i) – были ли мы в строке i (0 или 1)
- вектор zc(j) – были ли мы в столбце j (0 или 1)
- матрица M(i,j) – из какой ячейки мы пришли в ячейку (i,j) – хранится два int

Шаг 2 – переназначение без изменения потенциалов



Как эти метания оформить

Рекомендую использовать для хранения cv::Mat M(N, N, CV_32SC2, cv::Scalar(-1,-1)) – здесь «**CV_32SC2**» означает, что в каждой ячейке хранятся 2 числа int

- в c(j) – были ли мы в столбце j (0 или
- матрица M(i,j) – из какой ячейки мы пришли в ячейку (i,j) – хранится два int

Шаг 2 – переназначение без изменения потенциалов

Win!

	j=0	j=1	j=2	j=3	j=4
i=0	C		F,C		C
i=1	F,C				
i=2			C		
i=3		C			F,C

Матрица M для данного случая:

	j=0	j=1	j=2	j=3	j=4
i=0			(2,2)		(0,2)
i=1					
i=2			(2,-1)		
i=3		(3,4)			(0,4)

Шаг 2 – переназначение без изменения потенциалов

Оформим алгоритм наших «метаний» в поиске переназначения как рекурсивную функцию, которая

- получает начальную ячейку и рекурсивно обходит матрицу C, заполняя вектора zr(i), zc(j) и матрицу M
- если удалось найти свободную работу, на которую можно переназначить какого-то из сотрудников, то эта функция возвращает конечную ячейку обхода,
  -- тогда, возвращаясь по матрице M, мы сможем восстановить всю цепочку.
- если же мы ничего не нашли, функция возвращает (-1,-1)

Шаг 2 – переназначение без изменения потенциалов

```
pair<int,int> МЕТАТЬСЯ_И_ЗАПОЛНЯТЬ_МАТРИЦУ_M(prev_i, prev_j):
        i = prev_i
        для всех j от 1 до N таких, что C(i,j) == 1 и zc[j] == 0 и j != prev_j:
                M(i,j) := (prev_i, prev_j) // переход по горизонтали
                zc[j] := 1
                найти new_i такой, что F(new_i, j) == 1
                если такой new_i не нашелся:
                        вернуть (i, j)
                zr[new_i] := 1
                M(new_i, j) := (i,j) // переход по вертикали

                (res_i, res_j) := МЕТАТЬСЯ_И_ЗАПОЛНЯТЬ_МАТРИЦУ_M(new_i, j)

                если res_i >= 0 && res_j >= 0:
                        вернуть (res_i, res_j)
                // если res_i или res_j отрицательные, то --  переход к следующему кандидату j
        вернуть (-1, -1)
```

Шаг 2 – переназначение без изменения потенциалов

- При вызове функции МЕТАТЬСЯ_И_ЗАПОЛНЯТЬ_МАТРИЦУ_М мы сохраняем матрицу М и вектора zr(i) и zc(j).
- Если мы нашли цепочку, по которой можно сделать переназначение – переназначаем, и пере-инициализируем и матрицу М и вектора zr(i) и zc(j)
- Если же мы **не** нашли такую цепочку, то **не трогаем** zr(i) и zc(j) – они будут использованы при изменении потенциалов

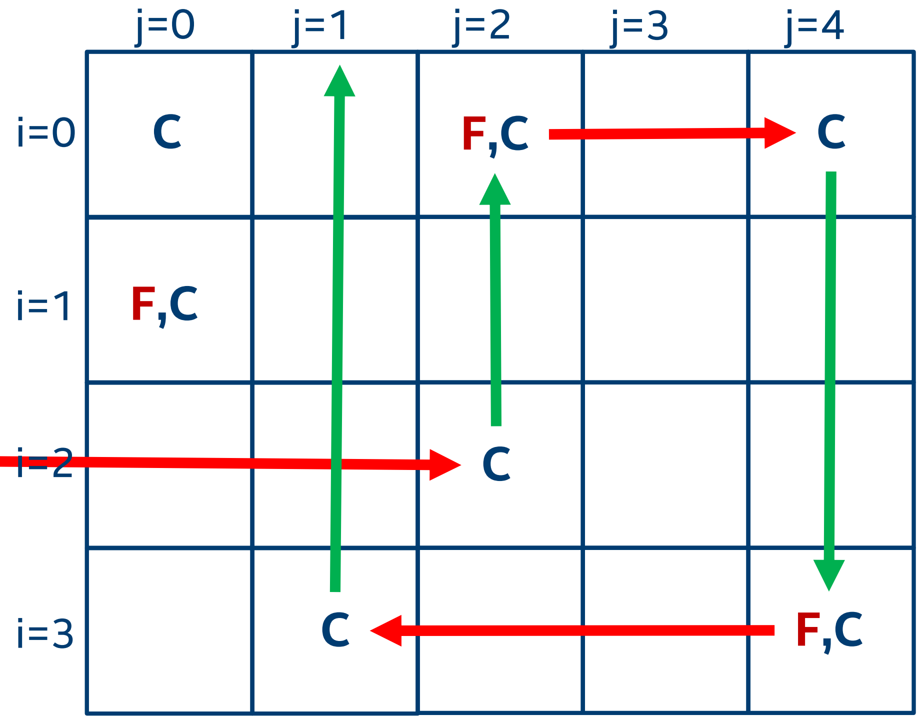
Шаг 2 – переназначение без изменения потенциалов

```
ПОПРОБОВАТЬ_ПЕРЕНАЗНАЧИТЬ_БЕЗ_ИЗМЕНЕНИЯ_ПОТЕНЦИАЛОВ():
    should_continue = true
    пока should_continue:
        should_continue = false
        заполнить матрицу M значениями (-1, -1)
        заполнить вектора zr и zc нулями
        для всех i от 1 до N:
            если в строке i уже есть F(i,j) == 1: // в строке есть назначение
                перейти к следующему i

            (res_i, res_j) := МЕТАТЬСЯ_И_ЗАПОЛНЯТЬ_МАТРИЦУ_M(i, -1)

            если res_i >= 0 && res_j >= 0:
                СДЕЛАТЬ_ПЕРЕНАЗНАЧЕНИЕ(res_i, res_j)
                заполнить матрицу M значениями (-1, -1)
                заполнить вектора zr и zc нулями
                should_continue = true // мы сделали переназначение, но может быть можно еще
```

Шаг 2 – переназначение без изменения потенциалов

```
void СДЕЛАТЬ_ПЕРЕНАЗНАЧЕНИЕ(res_i, res_j):
        i := res_i
        j := res_j
        пока i >= 0 && j >= 0:
                F(i,j) := 1 - F(i,j)
                (next_i, next_j) := M(i,j)
                i := next_i
                j := next_j
```

Шаг 2 – переназначение без изменения потенциалов



Запомним еще одно важное свойство наших метаний:
если мы зашли в какой-то столбец j, то мы переходим «по вертикали» в ту строку i в этом столбце, в которой F(i,j) = 1

Это значит, что если после метаний

- для некоторого j^* имеет место $zc(j^*) = 1$,
- и есть такое i^* что $F(i^*,j^*) = 1$
- то и $zr(i^*) = 1$

Общая схема алгоритма задачи о назначениях (rep)

```
void НАЙТИ_НАЗНАЧЕНИЕ():
        ИНИЦИАЛИЗИРОВАТЬ_ПОТЕНЦИАЛЫ()
        ЗАПОЛНИТЬ_МАТРИЦУ_КАНДИДАТОВ()
        ЖАДНАЯ_ИНИЦИАЛИЗАЦИЯ_F()
        если в каждой строке i есть ячейка F(i,j) == 1:
              выход // нашли решение жадным алгоритмом

        повторять:
           ЗАПОЛНИТЬ_МАТРИЦУ_КАНДИДАТОВ()
           ПОПРОБОВАТЬ_ПЕРЕНАЗНАЧИТЬ_БЕЗ_ИЗМЕНЕНИЯ_ПОТЕНЦИАЛОВ()

           // мы переназначили без изменения потенциалов все, что могли
           если в каждой строке i есть ячейка F(i,j) == 1:
                 выход

           // если оказались здесь, значит не смогли назначить работы
           // всем сотрудникам без изменения потенциалов
           ИЗМЕНИТЬ_ПОТЕНЦИАЛЫ()
```

Шаг 3 – изменение потенциалов

Иногда назначить всем сотрудникам задачи невозможно без изменения потенциалов.

Пример: если матрица А имеет вид

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 2 & 3 \\ 0 & 2 & 5 \end{pmatrix}$$

Тогда на первом шаге, при инициализации потенциалов мы получим

$$pr(i) = \min\{a(i,j), j = 1,2,3\}$$
$$pr = (0, 0, 0)$$
$$pc = (0, 0, 0)$$

	j=0	j=1	j=2
i=0		C	
i=1	C		
i=2	C		

Шаг 3 – изменение потенциалов

Как именно будем изменять потенциалы?

- Мы хотим сделать так, чтобы после изменения потенциалов мы могли бы использовать ранее найденную матрицу назначений F(i,j)
- Но мы ищем матрицу F(i,j) так, чтобы F(i,j) == 1 только если C(i,j) == 1
  А матрица C(i,j) == 1 тогда и только тогда, когда $a(i,j) - pr(i) - pc(j) = 0$
  – это может измениться после изменения потенциалов
- Значит, наше требование к изменению потенциалов:

  В тех ячейках, в которых F(i,j) = 1, значение $a(i,j) - pr(i) - pc(j)$
  не должно увеличиваться

Шаг 3 – изменение потенциалов

Мы сохранили вектора zr(i) и zc(j), которые заполняли в процессе «метаний»

Воспользуемся этими значениями, и найдем следующую величину:

d – минимальное значение $a(i,j) - pr[i] - pc[j]$
среди всех ячеек (i,j) таких, что zr[i] == 1 и zc[j] == 0

$$d = \min\{a(i,j) - pr[i] - pc[j]: \forall(i,j) \mathbin{/} zr[i] == 1, zc[j] == 0\}$$

Тогда d > 0

Доказательство: в процессе «метаний» мы бегали по всем строкам в которых zr(i) == 1, и если мы не заходили в какой-то столбец j, это значит, что в этих строках в этом столбце не было кандидатов, то есть

$$a(i,j) - pr(i) - pc(j) > 0 \text{ для всех } (i,j) \text{ таких, что } zr(i) == 1, zc(j) == 0$$

Шаг 3 – изменение потенциалов

Изменим потенциалы pr(i) и pc(j) так:

- для всех строк i таких что zr[i] == 1 мы прибавляем к pr[i] значение d
- для всех столбцов j таких, что zc[j] == 1 мы вычитаем из pc[j] значение d

Докажем, что так сделать можно – что после этого изменения pr(i) и pc(j) будут потенциалами.

Для этого нужно доказать, что и после изменений будет иметь место

$$a(i,j) - pr(i) - pc(j) \geq 0$$

Как изменится это значение?

Шаг 3 – изменение потенциалов

Изменим потенциалы pr(i) и pc(j) так:

- для всех строк i таких что zr[i] == 1 мы **прибавляем** к pr[i] значение d
- для всех столбцов j таких, что zc[j] == 1 мы **вычитаем** из pc[j] значение d

Тогда

- если (i,j) такие, что zr[i] == 0 и zc[j] == 0 – значение $a(i,j) - pr(i) - pc(j)$ **не изменится**
- если (i,j) такие, что zr[i] == 1 и zc[j] == 1 – значение $a(i,j) - pr(i) - pc(j)$ **не изменится** (мы прибавили и вычли d)
- если (i,j) такие, что zr[i] == 0 и zc[j] == 1 – значение $a(i,j) - pr(i) - pc(j)$ **увеличится** на d (pr[i] не изменится, а pc[j] уменьшится на d)
- если (i,j) такие, что zr[i] == 1 и zc[j] == 0 – значение $a(i,j) - pr(i) - pc(j)$ **уменьшится на d** (pr[i] увеличится на d, а pc[j] не изменится)
  – но мы выбрали d как **минимум** из таких чисел, значит $a(i,j) - wr(i) - wt(j) \geq 0$

Шаг 3 – изменение потенциалов

Изменим потенциалы pr(i) и pc(j) так:

- для всех [i] значение d
- для всех pc[j] значение d

Вопрос: как изменится матрица C?

Напомним: наше требование, чтобы в тех ячейках, в которых F(i,j) = 1, значение $a(i,j) - pr(i) - pc(j)$ **не должно увеличиваться**

Тогда

- если (i,j) та $- pr(i) \; - pc(j)$ **не изменится**
- если (i,j) та $- pr(i) \; - pc(j)$ **не изменится** (мы приба
- если (i,j) такие, что zr[i] == 0 и zc[j] == 1 – значение $a(i,j) \; - pr(i) \; - pc(j)$ **увеличится** на d (pr[i] не изменится, а pc[j] уменьшится на d)
- если (i,j) такие, что zr[i] == 1 и zc[j] == 0 – значение $a(i,j) \; - pr(i) \; - pc(j)$ **уменьшится на d** (pr[i] увеличится на d, а pc[j] не изменится)
  – но мы выбрали d как **минимум** из таких чисел, значит $a(i,j) - wr(i) - wt(j) \geq 0$

Шаг 3 – изменение потенциалов

Изменим потенциалы pr(i) и pc(

- для всех строк i таких что z … pr[i] значение d
- для всех столбцов j таких, ч … pc[j] значение d

Здесь значение C(i,j)
не изменится

Тогда

- если (i,j) такие, что zr[i] == 0 и zc[j] == 0 – значение $a(i,j) \; - pr(i) \; - pc(j)$ **не изменится**
- если (i,j) такие, что zr[i] == 1 и zc[j] == 1 – значение $a(i,j) \; - pr(i) \; - pc(j)$ **не изменится** (мы прибавили и вычли d)
- если (i,j) такие, что zr[i] == 0 и zc[j] == 1 – значение $a(i,j) \; - pr(i) \; - pc(j)$ **увеличится** на d (pr[i] не изменится, а pc[j] уменьшится на d)
- если (i,j) такие, что zr[i] == 1 и zc[j] == 0 – значение $a(i,j) \; - pr(i) \; - pc(j)$ **уменьшится на d** (pr[i] увеличится на d, а pc[j] не изменится)
  – но мы выбрали d как **минимум** из таких чисел, значит $a(i,j) - wr(i) - wt(j) \geq 0$

Шаг 3 – изменение потенциалов

Изменим потенциалы pr(i) и pc(j) так:

- для всех строк i таких что zr[i] == 1 мы **прибавля**
- для всех столбцов j таких, что zc[j] == 1 мы **вычит**

Тогда

- если (i,j) такие, что zr[i] == 0 и zc[j] == 0 – зна
- если (i,j) такие, что zr[i] == 1 и zc[j] == ение
  (мы прибавили и вычли d)
- если (i,j) такие, что zr[i] и zc[j] == 1 – значение $a(i,j) \ - pr(i) \ - pc(j)$
  увеличится на d [i] не изменится, а pc[j] уменьшится на d)
- если (i,j) такие, что zr[i] == 1 и zc[j] == 0 – значение $a(i,j) \ - pr(i) \ - pc(j)$
  уменьшится на d (pr[i] увеличится на d, а pc[j] не изменится)
  – но мы выбрали d как **минимум** из таких чисел, значит $a(i,j) - wr(i) - wt(j) \geq 0$

Здесь появятся новые ячейки такие, что C(i,j) = 1 (как минимум одна!
– та, в которой нашли минимум d)

Шаг 3 – изменение потенциалов

Изменим потенциалы pr(i) и pc(j) так:

- для всех строк i таких что zr[i] == 1 мы **прибавляем** к pr[i] значение d
- для все

А здесь C(i,j) станут равны нулю!

Но – как мы говорили ранее, если zc(j*) = 1 и F(i*,j*) = 1, то zr(i*) = 1.

Значит, если zr[i] == 0 и zc[j] == 1, то здесь не может быть F(i,j) = 1

Тогда

- если (i,j)
- если (i,j)
  (мы прибавили и выч
- если (i,j) такие, что zr[i] == 0 и zc[j] == 1 – значение $a(i,j)\ - pr(i)\ - pc(j)$ **увеличится** на d (pr[i] не изменится, а pc[j] уменьшится на d)
- если (i,j) такие, что zr[i] == 1 и zc[j] == 0 – значение $a(i,j)\ - pr(i)\ - pc(j)$ **уменьшится на d** (pr[i] увеличится на d, а pc[j] не изменится)
  – но мы выбрали d как **минимум** из таких чисел, значит $a(i,j) - wr(i) - wt(j) \geq 0$

Шаг 3 – изменение потенциалов

```
void ИЗМЕНИТЬ_ПОТЕНЦИАЛЫ():
        d = FLOAT_MAX // какое-то заведомо большое число
        для всех i от 1 до N таких, что zr[i] == 1:
              для всех j от 1 до N таких, что zc[j] == 0:
                   d := min(d, a[i,j] - pr[i] - pc[j])

        для всех i от 1 до N таких, что zr[i] == 1:
              pr[i] := pr[i] + d

        для всех j от 1 до N таких, что zc[j] == 1:
              pc[j] := pc[j] – d
      ЗАПОЛНИТЬ_МАТРИЦУ_КАНДИДАТОВ()
```

Общая схема алгоритма задачи о назначениях (rep)

```
void НАЙТИ_НАЗНАЧЕНИЕ():
        ИНИЦИАЛИЗИРОВАТЬ_ПОТЕНЦИАЛЫ()
        ЗАПОЛНИТЬ_МАТРИЦУ_КАНДИДАТОВ()
        ЖАДНАЯ_ИНИЦИАЛИЗАЦИЯ_F()
        если в каждой строке i есть ячейка F(i,j) == 1:
                выход // нашли решение жадным алгоритмом

        повторять:
            ЗАПОЛНИТЬ_МАТРИЦУ_КАНДИДАТОВ()
            ПОПРОБОВАТЬ_ПЕРЕНАЗНАЧИТЬ_БЕЗ_ИЗМЕНЕНИЯ_ПОТЕНЦИАЛОВ()

            // мы переназначили без изменения потенциалов все, что могли
            если в каждой строке i есть ячейка F(i,j) == 1:
                    выход

            // если оказались здесь, значит не смогли назначить работы
            // всем сотрудникам без изменения потенциалов
            ИЗМЕНИТЬ_ПОТЕНЦИАЛЫ()
```