

## 18. Операции над множествами (масками): объединение, пересечение, разность. Реализация в программе. Проверка, что одна маска является подмножеством другой. Проверка, что число является степенью двойки.

Идея ДП по маскам: пусть  $n$  какое-то небольшое число ( $\leq 30-60$ ), тогда мы можем эффективно закодировать все подмножества множества  $\{0, \dots, n-1\}$ . Для этого рассматриваем маску. Она будет состоять из  $n$  символов, каждый из которых - единица или ноль. Единица на  $i$ -м месте в маске означает, что мы берем число  $i$  в подмножество, 0 - не берем в подмножество

### Операции над масками

Пусть даны два множества  $A$  и  $B$  с соответствующими им масками  $mask_A$  и  $mask_B$

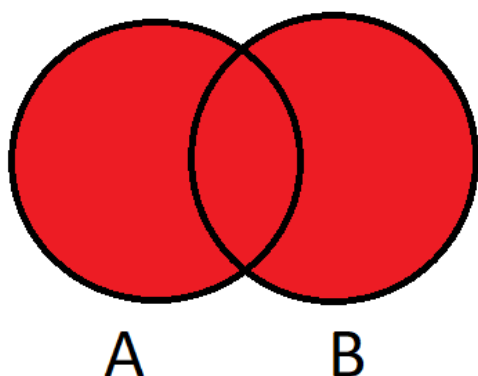
**Объединение**

$$A \cup B = mask_A \mid mask_B$$

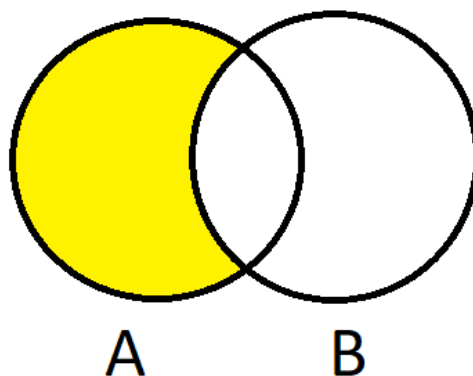
**Пересечение**

$$A \cap B = mask_A \& mask_B$$

**Разность**  $A \setminus B = (mask_A \mid mask_B) \oplus mask_B$ .



$$mask_A \mid mask_B$$



$$(mask_A \mid mask_B) \oplus mask_B$$

### Проверка, что одна маска является подмножеством другой

Пусть даны две маски  $A$  и  $B$ . Утверждается, что  $B \subset A \iff (A \& B) = B$

▲

→ Пусть  $B \subset A$ , тогда если в маске  $B$  на месте  $i$  стоит единица, то и в маске  $A$  на месте  $i$  стоит единица, а значит, их побитовое "и" даст в точности  $B$ .

← Если  $(A \& B) = B$ , то в маске  $A$  единицы стоят как минимум на тех местах, что и в маске  $B$ , иначе их побитовое "и" не дало бы единицу на том же месте, что и в маске  $B$ . Это значит, что  $B \subset A$  ■

### Проверка, что число является степенью двойки.

Утверждается, что число  $a$  является степенью двойки  $\iff a \& (a-1) = 0$

▲

→ Рассмотрим двоичную запись числа  $a$ . Если  $a = 2^n$ , то  $a = 1\underbrace{0\dots0}_n \Rightarrow a - 1 = \underbrace{1\dots1}_{n-1} \Rightarrow a \& (a - 1) = 0$   
 ← Пусть  $a \neq 1\underbrace{0\dots0}_n$ , тогда  $a = \underbrace{1}_n \dots \underbrace{1}_{i-1} \dots$ , т.е. существует единица на  $i$ -м месте в записи числа  $a$ , тогда  $a - 1 = \underbrace{1}_n \dots$ , т.е. старший бит не обнуляется, а значит, побитовое "и" не может равняться нулю. Противоречие. ■

## 19. Задача о самом дешёвом гамильтоновом пути: решение за $O(2^n n^2)$

Запишем важную и полезную функцию, которая позволяет извлекать  $i$ -й элемент маски:

```
bool bit(long long mask, int pos) {
    return (mask >> pos) & 1;
}
```

### Формулировка задачи

Дан полный, неориентированный, взвешанный граф.

*Гамильтонов путь в графе* - это путь, который начинается в какой-то вершине графа, проходит по всем вершинам и посещает все вершины графа ровно по одному разу

Среди всех таких путей нас интересует путь минимальной стоимости

Пусть у нас есть какой-то путь, проходящий по каким-то вершинам графа ровно один раз и заканчивающийся в вершинке  $v$ . Тогда для того, чтобы продолжить этот путь, нам необходимо знать, в какой вершинке мы находимся и маску посещенных вершин. Отсюда возникает  $dp$ :

1.  $dp[v][mask]$  - это стоимость минимального пути, который где-то начинается, посещает все вершины маски ровно по одному разу и заканчивается в  $v$
2. База: Пути длины 0. Когда мы стоим в какой-то вершинке и еще никуда не пошли.  $dp[v][2^v] = 0$ , все остальные значения положим  $+\infty$
3. Переход: перебираем все ребра, исходящие из  $v$ , которые ведут в вершины, еще не посещенные (в маске на месте соответствующей вершины будет стоять 0), и смотрим, куда можем пойти.

$(1 \ll u) | mask$  - взять побитовое "или" числа  $2^u$  и  $mask$ . Нетрудно заметить, что такая процедура просто ставит единичку на  $u$ -й бит маски

```
for (int mask = 0; mask < 2^n; ++mask) {
    for (int v = 0; v < n; ++v) {
        for (int u = 0; u < n; ++u) {
            if (bit(mask, u)) continue;
            int newmask = (1 << u) | mask;
            dp[u][newmask] = max(dp[u][newmask], dp[v][mask] + cost[v][u]);
        }
    }
}
```

4. Ответ: Минимальное  $dp[u][2^n - 1]$ . То есть нас интересуют все пути, которые посещают все вершины, выбираем из них тот, что имеет минимальную стоимость

### Асимптотика

За счет циклов  $O(2^n n^2)$

## 20. Задача о максимальной клике: решения за $O(2^n n^2)$ , $O(2^n n)$ , $O(2^n)$

### Формулировка задачи

Дан неориентированный, взвешанный граф.

*Клик*ой в неориентированном графе называется подмножество вершин, каждые две из которых соединены ребром графа.

Хотим найти максимальную по мощности клику в данном графе

$O(2^n n^2)$

Перебираем все подмножества  $n$  и за  $n^2$  перебором проверяем, что это клика

$O(2^n n)$

1. Положим  $dp[mask] \begin{cases} true, mask - klika \\ false, else \end{cases}$

2. Будем делать дп назад

3.  $dp[mask]$  - клика  $\iff$  для любой вершины  $v$  из этой маски выполнено:

1  $dp[mask \oplus 2^v] = true$ .  $mask \oplus 2^v$  - на  $v$ -ое место маски ставит false. Здесь мы проверяем, что маска без этой вершинки  $v$  - клика

2  $v$  соединена со всеми вершинами маски

4. База:  $dp[0] = true$

5. Ответ: максимальная по мощности маска, для которой  $dp[mask] = true$

### Асимптотика

Перебор всех масок:  $2^n$ . Для каждой маски необходимо за линейное время определить какую-нибудь вершинку, которая входит в эту маску (просто пройтись по  $n$  битам и найти тот, для которого  $bit(mask, i) = true$ ). Далее идем по маске и за линейное время определяем, связана ли вершинка, которую мы выбрали, со всеми остальными. Итого получаем  $O(2^n n)$

$O(2^n)$

Новый алгоритм - это оптимизация предыдущего.

1 Для каждой вершинки  $u$  посчитаем маску ее соседей. Пусть  $neigh[u]$  - маска соседей  $u$ . Сделаем это на этапе ввода графа.

2 Теперь бит  $v$  выбираем не произвольным образом. Это будем старший включенный бит маски. Почему старший? Его будет удобно насчитывать, что мы поймем дальше

Тогда проверка пункта 3.2(см. предыдущий алгоритм) будет проверяться за  $O(1)$ :

$dp[mask \oplus 2^v] \subset neigh[v]$ . Из 18 билета мы знаем, что  $a \subset b \iff (a \& b) = a$ . Значит, проверка:  $(dp[mask \oplus 2^v] \& neigh[v]) = dp[mask \oplus 2^v]$

Теперь поймем, как хранить старший бит

```
int oldest_bit = -1;
for(int mask = 1; mask < 2^n; ++mask){
    if(mask & (mask - 1) == 0) ++oldest_bit;
    ...
}
```

Сначала отметим старший бит равным -1. Очевидно, что старший бит маски будет меняться только в том случае, если мы переходим через степень двойки ( $mask \& (mask - 1) == 0$  - проверка на то, что  $mask$  - степень двойки, см. 18 билет)

### Асимптотика

Таким образом, мы научились делать переход за  $O(1)$ . Значит, итоговая асимптотика  $O(2^n)$

## 21. Подсчёт всех значений $b(mask) = \max_{s \subset mask} a(s)$ для данного набора значений $a(0), a(1) \dots a(2^n - 1)$ за $O(2^n n)$

Мы хотим для каждой из масок  $0 \dots 2^n - 1$  посчитать максимальное по мощности подмножество маски, являющееся кликой. Пусть  $a(V) =$

0, если  $V$  - не клика

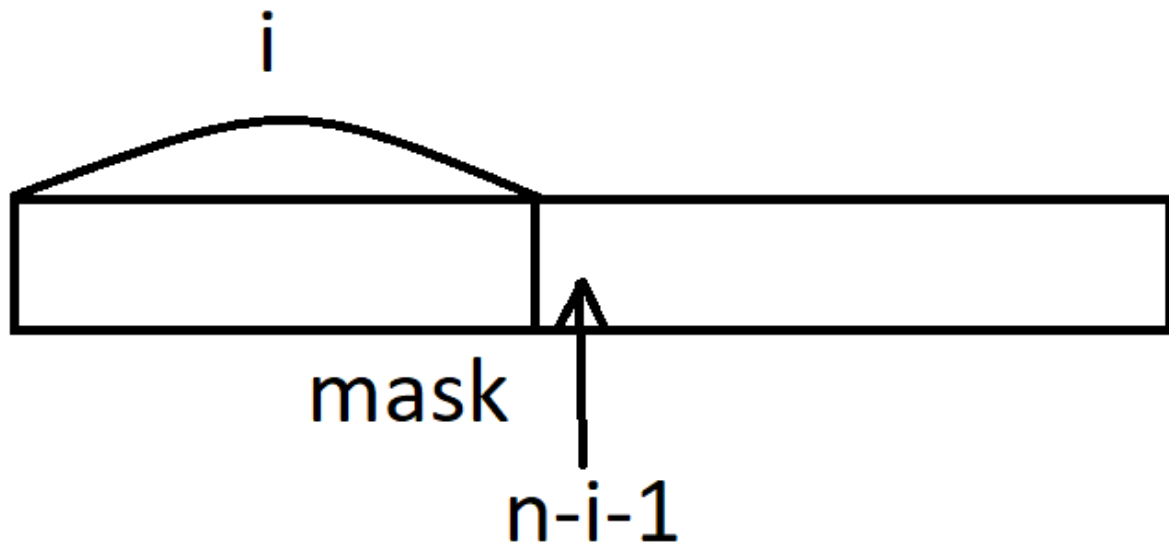
$|V|$ , если  $V$  - клика

Тогда мы хотим найти  $b(mask) = \max_{V \subset mask} a(V)$ . Будем делать это с помощью дп

1.  $dp[k][mask] = \max_{V \subset mask} a(V)$ , где  $V$  - такое, что первые  $k$  битов  $V$  и  $mask$  совпадают

2. База  $dp[n][mask] = a(mask)$

3. Переход. Будем ходить по подмаскам. Пусть нам известно  $dp[i+1][mask]$ . Научимся переходить к  $dp[i][mask]$ . Зафиксируем старшие  $i$  битов, тогда номер у следующего, если считать от начала маски,  $n-i-1$



1 Если  $\text{!bit}(mask, n - i - 1)$ , то при переходе к подмаске мы не убрали ни одного элемента (т.к. при переходе к подмаске 0 не может стать единицей, значит, остается только 0), а значит, максимальная мощность подмножества, являющегося кликой, не изменилась,  $dp[i][mask] = dp[i+1][mask]$

2 Если  $\text{bit}(mask, n - i - 1)$ , то у нас есть выбор, при переходе к подмаске оставить единицу на этом месте или превратить ее в ноль. Так как мы ищем максимальную мощность, то  $dp[i][mask] = \max(dp[i+1][mask], dp[i][mask \oplus 2^{n-i-1}])$

4. Ответ:  $b(mask) = dp[0][mask]$

### Асимптотика

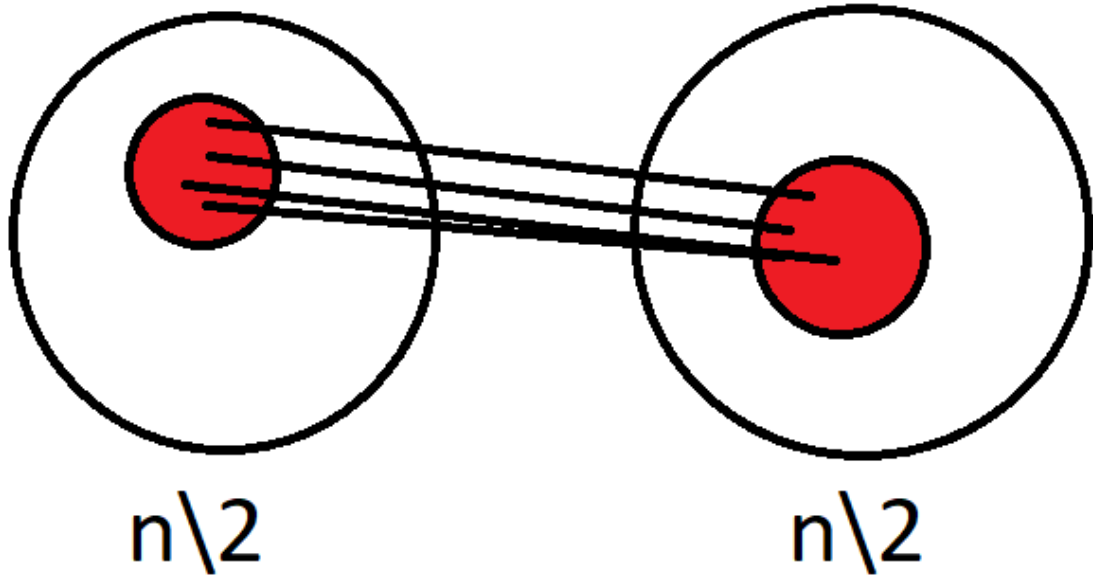
1. Базу можно насчитать за  $O(2^n)$

2. Перебираем все  $i = n-1 \dots 0$ , перебираем все маски, делаем переход за  $O(1)$

Итог:  $O(2^n n)$

## 22. Задача о максимальной клике: решение за $O(2^{\frac{n}{2}}n)$ .

Здесь будет использоваться идея meet-in-the-middle. Она заключается в том, что мы делим  $n$  пополам и получаем две маски длины  $\frac{n}{2}$



Очевидно, что если маска является кликой, то, если мы распилим ее пополам, новые маски останутся тоже кликами.

Если мы нашли какую-то клику в одном из множеств, то, если мы дополним ее нулями, получится клика во множестве масок длины  $n$ .

Отсюда получаем, что все маски длины  $n$  можно разделить на 3 группы

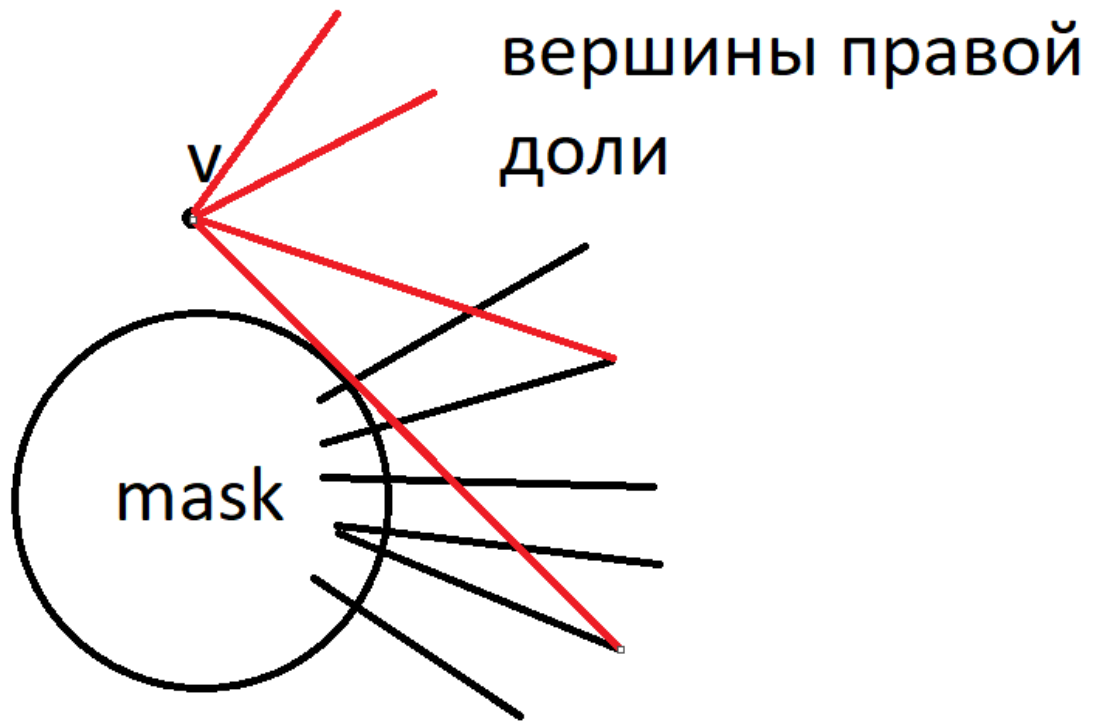
- 1 Правая часть - нули, левая - клика в левом множестве
- 2 Левая часть - нули, правая - клика в правом множестве
- 3  $u$  (левая часть маски длины  $n$ ) - клика в левом множестве,  $v$  (правая часть маски длины  $n$ ) - клика в правом множестве

### Алгоритм

Рассмотрим маску вершин левой доли  $mask$ . И запишем маску  $mask\_right$  вершин правой доли, которые соединены со всеми вершинами из  $mask$ . (То есть перебираем все вершины правой доли, и если какая-то вершина соединена со всеми из  $mask$ , добавляем ее в маску  $mask\_right$ )

Пусть  $dp1[mask] = mask\_right$ ,  $neigh'[v]$  - маска соседей  $v$  в правой доле (вершина  $v$  из левой доли), посчитать все  $neigh'[i]$  можно за квадрат простым перебором

1.  $dp1[0]$  - вся правая доля
2. Хотим добавить в маску вершинку  $v$ .  $dp1[mask|2^v] = (dp1[mask] \& neigh'[v])$



В маску  $dp1[mask|2^v]$  входит пересечение соседей  $v$  и соседей всех вершин  $mask$ . По определению пересечение всех соседей  $mask$  - это  $dp1[mask]$

Зафиксируем множество  $U$ , являющееся кликой в левой доле. Тогда дополнение  $V$  множества  $U$  до клики во всем множестве - это некоторое подмножество  $dp1[U]$ . Поскольку  $U$  - клика, и из  $U$  проведены все ребра в вершины правой доли, то  $V$  - это максимальное подмножество  $dp1[U]$ , которое является кликой (так как все остальные вершины соединены между собой, осталось найти множество вершин из  $dp1[U]$ , которые соединены между собой).

Для любой маски  $mask$  правой доли найдем максимальное по мощности множество  $V \subset mask$ , являющееся кликой. Будем это делать, используя алгоритм из билета 21, за  $2^{\frac{n}{2}}n$

#### Асимптотика

1. Насчитать  $dp1[mask]$  - перебор всех масок за  $2^{\frac{n}{2}}$ , для каждой маски проходимся по всем вершинам левой доли от  $0 \dots \frac{n}{2}$ , проверяем, входит ли вершина в  $mask$   $O(1)$ , если не входит, то делаем переход по формуле за  $O(1)$ . Тогда этот шаг за  $O(2^{\frac{n}{2}}n)$
2. Насчитать для каждой маски  $mask$  правой доли максимальную клику  $a[mask]$ , являющуюся подмаской за  $O(2^{\frac{n}{2}}n)$
3. Перебираем все маски  $mask$  левого множества, являющиеся кликами, для каждого  $dp1[mask]$  максимальная клика в правом множестве -  $a[dp1[mask]]$ , тогда искомое множество - объединение  $mask$  и  $a[dp1[mask]]$ . Мощность объединения можно посчитать за  $O(n)$ , насчитывая количество единиц в двоичной записи масок. Третий шаг за  $O(2^{\frac{n}{2}}n)$

Итоговая асимптотика:  $O(2^{\frac{n}{2}}n)$

## 23. Симпатичные узоры: количество раскрасок таблицы $n \times m$ в два цвета без одноцветных квадратиков $2 \times 2$ . Прямой профиль: решения за $O(4^n(n+m))$ и $O(8^n \log m)$

$O(4^n(n+m))$

### Формулировка

Дана таблица  $n \times m$  и два цвета

1 - черный

2 - белый

Мы не хотим, чтобы в таблице были квадратики  $2 \times 2$ , раскрашенные в один цвет  
Сколько есть таких раскрасок?

Предположим, мы уже раскрасили  $j$  столбцов. Что нам нужно знать для продолжения раскраски? Очевидно, нам нужно знать только цвета последнего столбца. Отсюда возникает динамика:

1.  $dp[j][mask]$  - сколько есть раскрасок  $j$  столбцов так, чтобы последний из них был в точности покрашен в  $mask$
2. Переход: перебираем все маски, соответствующие раскраске следующего столбца и смотрим, не нарушается ли симпатичность узора

Предподсчитаем корректность перехода между масками. Для этого перебираем все комбинации масок и проверяем, сравнивая их за  $O(n)$ , корректен ли переход. асимптотика предподсчета  $O(4^n n)$

3.База:  $dp[1][mask] = 1$

4.Ответ: сумма по всем  $i$   $dp[m][i]$

```
for (int j = 1; j <= m - 1; ++j) {
    for (long long mask = 0; mask < 2^n - 1; ++mask) {
        for (long long mask1 = 0; mask1 < 2^n - 1; ++mask1) {
            if (ok[mask1][mask])
                dp[j + 1][mask1] += dp[j][mask];
        }
    }
}
```

### Асимптотика

За счет циклов  $O(4^n(m))$  + предподсчет  $O(4^n(m))$ . Итого  $O(4^n(m+n))$

$O(8^n \log m)$

Заметим, что значение  $dp[i][mask] = \sum_{mask1=0}^{2^n-1} ok[mask][mask1] * dp[i-1][mask1]$ , значит,

$$\begin{pmatrix} dp[i][0] \\ dp[i][1] \\ \dots \\ dp[i][2^n-1] \end{pmatrix} = \begin{pmatrix} ok[0][0] & ok[0][1] & \dots & ok[0][2^n-1] \\ ok[1][0] & ok[1][1] & \dots & ok[1][2^n-1] \\ \dots & \dots & \dots & \dots \\ ok[2^n-1][0] & ok[2^n-1][1] & \dots & ok[2^n-1][2^n-1] \end{pmatrix} * \begin{pmatrix} dp[i-1][0] \\ dp[i-1][1] \\ \dots \\ dp[i-1][2^n-1] \end{pmatrix}$$

$$= ok * \begin{pmatrix} dp[i-1][0] \\ dp[i-1][1] \\ \dots \\ dp[i-1][2^n-1] \end{pmatrix}$$

$$\text{Тогда} \begin{pmatrix} dp[m][0] \\ dp[m][1] \\ \dots \\ dp[m][2^n - 1] \end{pmatrix} = ok^{m-1} * \begin{pmatrix} dp[1][0] \\ dp[1][1] \\ \dots \\ dp[1][2^n - 1] \end{pmatrix}$$

### Асимптотика

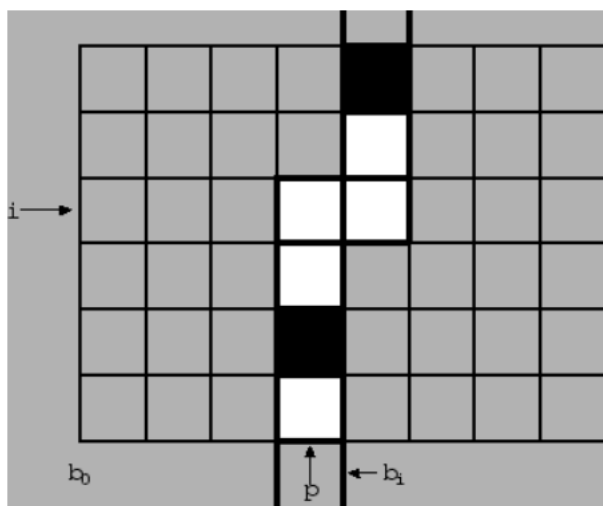
1. Матричное умножение  $O((2^n)^3 \log m) = O(8^n \log m)$

Это хорошая асимптотика, если  $n \leq 6$ ,  $m \leq 10^{100}$

## 24. Симпатичные узоры. Изломанный профиль: идея решения за $O(2^n mn)$

Формулировка задачи та же самая

Теперь мы режем нашу табличку не по прямой, а по ломанной прямой. Берем некоторое количество элементов в начале рассматриваемого столбца, а остальные - из предыдущего



Хотим сделать то же самое: предполагая, что все левее разреза покрашено корректно, знать, как будем продолжать красить

1.  $dp[j][i][mask]$  = сколько способов дойти до  $j$  ячейки  $i$  строки с маской  $mask$  у нас есть.  $(i,j)$  - излом,  $mask$  - соответственная маска вдоль этого излома (на картинке - раскрашенная полоска)
2. База  $dp[1][n][mask] = 1$ , все остальные - 0
3. Переход: Просчитываем, как можно изменить квадратик, стоящий на изломе (то есть он может быть 0 или 1). Если переход корректен, красим его в этот цвет, переходим к  $dp[j][i+1][newmask]$ , где новая маска пересчитывается в соответствии с тем, как мы покрасили клетку. Заметим, что хотябы один переход возможен всегда

Алгоритм:

- 1 Цикл по  $j$
- 2 Цикл по  $mask$
- 3 Цикл по  $i$
- 4 Просчитываем квадратик излома
  - Если можно поставить единичку или ноль, ставим, получаем  $newmask$ ,  $dp[j][i+1][newmask] += dp[j][i][mask]$  (если дошли до конца столбца, переходим в  $dp[j+1][1][newmask]$ )

Ответ: сумма по всем  $mask$   $dp[m][n][mask]$