26.01.2021

Графы

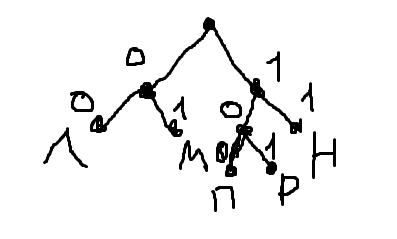


F = (x | y) & -(y = x) & -w



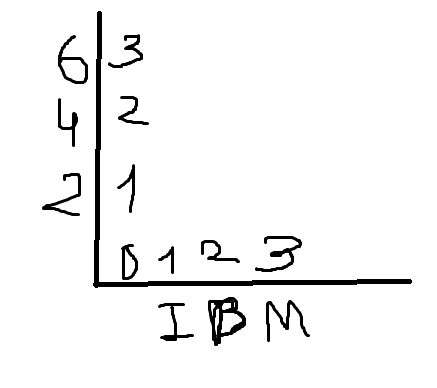
Л М Н П Р

00 01 11 min x

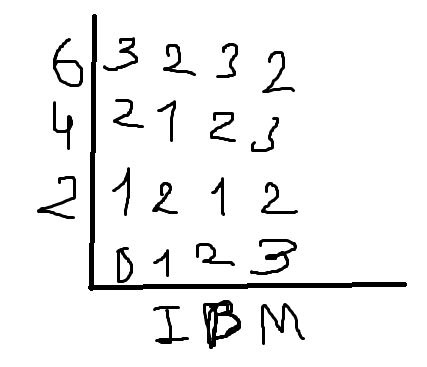


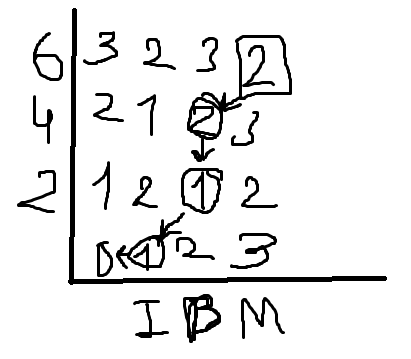
Динамическое программирование

Создаём матрицу (n + 1)x(m + 1), заполняем первые строку и столбец числами от 0 до n и m соответственно



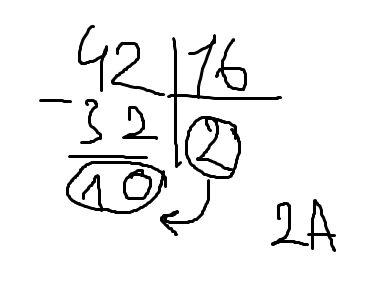
Обход с расчётом

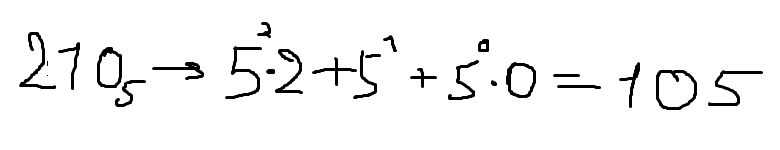


Восстановление маршрута(удаление – влево, вставка – вниз, ничего - диагональ)

02.02.2021

Системы счисления, алгоритмы перевода





(001)(101)(011)<2>, переводим тройки цифр в нужную систему

= 153<8>

49^7 + 7^21 – 7 сколько цифр 6 в <7>

7^0 = 1

7^1 = 10

7^2 = 100…

49^7 = 7^14 = 100 000 000 000 000<7>

100 000 000 000 000<7> + 7^21(1 000 000 000 000 000 000 000<7>) = 1 000 000 100 000 000 000 000<7>

1 000 000 100 000 000 000 000<7> – 7(10<7>) = 1 000 000 066 666 666 666 660<7> => кол-во 6 = 13

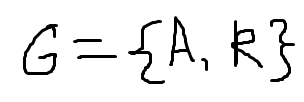
F(n) = 1 при n = 1

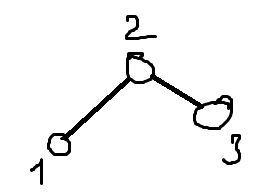
F(n) = n + F(n-1), если n – чётное

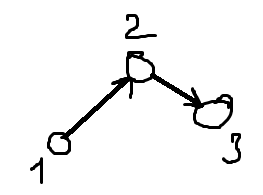
F(n) = 2\*F(n-2), если n > 1 и n – нечётное

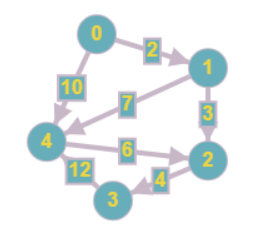


Графы



 - неориентированный

- ориентированный, если (i, k) & (k, j) => (i, j) – транзитивное замыкание



0, 2, 0, 0, 10,

0, 0, 3, 0, 7,

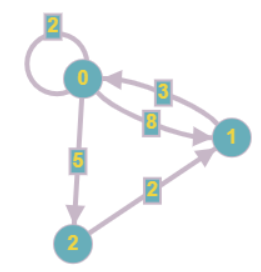
0, 0, 0, 4, 0,

0, 0, 0, 0, 12,

0, 0, 6, 0, 0,

D = [2, inf, inf, 10] -> (curr: 1) D = [2, inf, inf, 9] -> (curr: 1) D = [2, 5, inf, 9] -> (curr: 2) D = [2, 5, 9, 9] ->…

Дейкстра + транз. замыкание (алг. Флойда)



for()

for()

for()

if(M[i][j] > M[i][k] + M[k][j])

M[i][j] = M[i][k] + M[k][j];

09.02.2021

2 8 9 6 7 3 5 4 1

M[i] < M[i+1]

…[i+1][k-1]

16.02.2021

abcde -> edcba(“strreverse.cpp”)

Инверсионная пара: пара, где ni > nj, i < j

2 8 9 6 7 3 5 4 1

Таблица инверсии:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 0 | 4 | 5 | 4 | 2 | 2 | 0 | 0 |

(“invpair.c”)

5 1 3 4 2

02.03.2021

Линейный список – множество, сост. из n узлов, структурные свойства которого ограничены линейным относительным положением узлов.

Стек – добавить в начало, взять из начала.

Очередь – добавить в конец, взять из начала.

Дек (double-ended queue) – добавить в обе стороны, взять с обоих сторон.

struct list {

int data;

struct list \*next;

};

typedef struct stack {struct list \*top;} Stack;

void makenull(Stack \*S){

struct list \*p;

while(S->top){

p = S->top;

S->top = p->next;

free(p);

}

}