**Коридор з колонами**

**Ідея:**

Якщо ми радіус кожної колони збільшемо на D/2, ліву стіну зсунемо вправо на D/2, а праву – вліво, то всі «дірки» зменшаться в розмірах на D (якщо раніше в «дірку» влазив стіл максимум радіуса R, то тепер буде влазити максимум радіуса R – D/2). Припустимо стіл діаметром D не можна пронести по коридору. Тоді після таких перетворень дірок, через які не влазив стіл, не буде (нехай в них влазив стіл діаметру 2R < D. Після перетворень у них буде влазити стіл діаметру 2R-D <0. Тобто там, де були дірки, будуть перетинатись (саме перетинатись, а не дотикатись) колони між собою або зі стінами.). Припустимо можна пронести стіл максимум діаметру D. Це значить, що не існує шляху, де не було би дірки, куди влазить стіл максимального діаметру D. Після наших перетворень на місці тієї дірки буде саме дотик колон між собою або зі стінами. Тож ідея в тому, щоб зробити бінарний пошук по D, поки точність не буде більшою, ніж 10-6. Оскільки 1<=W<=100000, то за **x** ітерацій (перевірок), ми знайдемо відповідь.

**План:**

«В умі» збільшуємо радіуси всіх колон на певну величину D/2, праву стіну зсуваємо вліво на цю величину, а ліву вправо. Шукаємо (за O(N2)) відстань між центрами всіх колон і стінками та записуємо в матрицю (N+2 x N+2) «1», якщо перетинаються (саме перетинаються, а не дотикаються), і «0» в іншому випадку. Очевидно, матриця симетрична. Нехай нульовий рядочок (стовбець) – містить інформацію про дотик до лівої стіни, а (N+1) – про дотик до правої стіни. По цій матриці з’ясуємо, чи якось не заблоковано стінами і колонами прохід, тобто чи можна дійти тільки по колонам від лівої стіни до правої (чи нема такої послідовності одиничок в матриці, які приведуть від лівої до правої стіни). Пошук шляху будемо здійснювати рекурсивно (починаючи з нульового – в ньому міститься інформація, хто перетинається з лівої стіною):

В масиві (b) будемо зберігати інформацію про те, в яких рядках побували. (Щоб не попадати туди кілька разів. Якщо в i-му рядку були, то b[i] == 1.)

Перевіряємо останній елемент кожного рядка, в якому буваємо, бо він ідентифікує, чи є шлях до правої стіни (якщо «1», то шлях існує, повертаємо true). Йдемо по і-му рядочку матриці до N+1 не включно. Якщо на j-му елементі зустрічаємо «1», то перевіряємо, чи ми були в j-му рядку. Якщо не були, то переходимо на нього (запускаємо рекурсію з параметром j) і повторюємо попередні дії. Коли кінчився цикл, повертаємо false. Якщо функція повертає true – це значить, що коридор заблоковано і стіл діаметру D не можна пронести.

Псевдокод:

boolean check(int i) {

b[i] = 1;

if (a[i][N+1] == 1) {

return true;

}

for(int j = 1; j < N+1; j++) {

if (a[i][j] == 1) {

a[i][j] = 0;

a[j][i] = 0;

if (b[j] != 1) {

if (check(j)) {

return true;

}

}

}

}

return false;

}

Отже, якщо функція поверне true, то такий шлях існує, а значить, як мінімум, в одному місці, шлях заблоковано і стіл не можна пронести. Оскільки в рекурсії ми не більше, ніж кожний елемент матриці перевіряємо на 0 (за рахунок того, що буваємо максимум 1 раз в рядку), то таку перевірку в гіршому випадку зробимо за O(N2). Отже, алгоритм працює за

з поганою константою. Додаткова пам’ять виділяється на матрицю відстаней: O(N2).