בעיית הבקבוקים:

נתונים 2 בקבוקים עם קיבולות a,b.   
מתחילים על כמות מים 0 בכל בקבוק ובכל שלב מותר לבצע את אחת מהפעולות הבאות:  
1. למלא את אחד הבקבוקים. עד הסוף.  
2. לרוקן את אחד הבקבוקים. עד 0.  
3. להעביר מבקבוק אחד לשני עד שהשני מתמלא והייתרה נשארת בבקבוק הראשון.

**מטרה**: למצוא מסלול קצר ביותר המתחיל ממצב של 0,0 ומגיע למצב בו יש x,y (כלומר x מים בבקבוק הראשון ו y בבקבוק השני) או פשוט מסלול קצר בין כל 2 מצבים.

**אלגוריתם**:  
נבנה גרף בו כל קודקוד מייצג את המצב של 2 הבקבוקים. הצלעות בגרף יהיו המעברים המותרים.  
לאחר מכן, נשתמש ב FW ונוכל למצוא מסלולים קצרים ביותר בגרף הנ"ל.  
  
**סיבוכיות**: בניית הגרף: כאשר: . קריאה ל FW - . סה"כ:.

**פסאודו קוד:**

public static double[][] bottleProblem(int a, int b) {

int n = (a+1)\*(b+1);

int[][] graph = new int[n][n];

for (int i = 0; i <= a; i++) {

for (int j = 0; j <= b; j++) {

graph[index(i,j,b)][index(0,j,b)] = 1;

graph[index(i,j,b)][index(i,0,b)] = 1;

graph[index(i,j,b)][index(a,j,b)] = 1;

graph[index(i,j,b)][index(i,b,b)] = 1;

graph[index(i,j,b)][index(Math.max(0, i - (b-j)),Math.min(i + j, b),b)] = 1;

graph[index(i,j,b)][index(Math.min(i + j, a),Math.max(0, j - (a-i)),b)] = 1;

}

}

for (int i = 0; i < n; i++) {

graph[i][i] = 0;

}

double[][] ans = FW.FW(graph);

return ans;

}

public static double shortestpath(int a, int b, int i, int j, int x, int y) {

double[][] ans = bottleProblem(a,b);

return ans[index(i,j,b)][index(x,y,b)];

}

public static int index(int i, int j,int b) {

return i\*(b+1) + j;

}