שידוכים:

חד צדדיים (סטודנטים למעונות)

שוק דו צדדי (שידוכים, התאמה לפקולטות)

החלפה:

החלפת בתים (מעגלי המסחר)

כליות

מכרזים:

מחיר שני - פריט אחד

כמה פריטים (gsp, vcg) - שונים

כמה פריטים (gsp, vcg) - זהים

מיקסום רווח עבור המוכר במכרז:

מיירסון

בולוב קמפרר

מכרז דו צדדי:

VCG

מיירסון וסטיוארט

מקאפי

שיוויי משקל:

נאש

בייס - נאש

חלוקה:

עוגה ל2 אנשים (cut and choose)

3 אנשים ויותר לא קשיר (המפחית האחרון, אבן פז)

3 אנשים ויותר לא קשיר ללא קנאה (סלפידג' קונוויי)

3 אנשים קשיר ללא קנאה (הסכין המסתובבת)

n אנשים ללא קנאה (סימפלקסונים)

חלוקת עוגה עם ערך שלילי (תורנויות, שמירה, כיסוח דשא) - סימפלקסונים

חלוקה אמיתית - רק בתנאים ספציפיים - bey Huzhang..

יעילות

חלוקת דירה:

חפצים לא ניתנים לחלוקה - דירה (אורדינלי - סו, קרדינלי- מקסום סכום הערכים+הונגרי)

קביעת מחירים לחדרים

בעיית הטרמפיסט

חלוקת חפצים בדידים בין אנשים ללא קנאה:

המנצח המתוקן - מחלקים את החפץ

חלוקה הוגנת בקירוב (EF1)

מעגלי הקנאה

מקסום מכפלת הערכים

ביטקוין

```
void cutAndChoose(Agent a, Agent b){

        double halfA = a.mark(a.eval(1)/2);
        double halfB = b.mark(b.eval(1)/2);

        if(halfA<halfB){
                print("Agent a recieves [0,"+halfA+"].
                        Agent b recieves ["+halfA+",1].");

        if(halfA>halfB){
                print("Agent b recieves [0,"+halfA+"].
                        Agent a recieves ["+halfA+",1].");
}
```

<u>להוכיח שמנגנון אינו אמיתי:</u>

להביא דוגמה נגדית כך שע"י שקר המשקר מרוויח.

<u>להוכיח שמנגנון אמיתי:</u>

להראות שהרווח המקסימלי מתקבל לכל היותר גם כשדובר אמת.

```
double 1/n (Agent a, int n){            //פונקציה המחזירה את החלק היחסי עבור המשתתף
        return a.mark(a.eval(1)/n);
}


double interval = 0;            //קטע העוגה שכבר חולקה
int agent=0;                    //המשתתפים שכבר הוצאו מהמשחק


void lastDiminisher(Agent [] agents){
        int n = agents.size;            //מספר משתתפים

        if(n==1){               //האחרון שנשאר מקבל אינטרוול עד 1
                print("Agent "+(++agent)+" receives ["+interval+",1]. ");
                return;
        }

        int index;
        double min = 1/n(agents[0], n);         //ההצעה הנמוכה ביותר min

        for(int i=1; i<n; i++){
                double corrent=1/n(agents[i],n);
                if(corrent<min){
                        min=corrent;            //נברור את המציע הנמוך ביותר מבין אילו שנשארו
                        index=i;                //נשמור את המציע הנמוך כדי שנוכל להוציא אותו בסיבוב הבא
                }
        }
        print("Agent "+(++agent)+" receives ["+interval+", "+(interval=+min)+"]. ");

        swap(agents[index], agents[n-1]);       //זריקת המפחית האחרון לתא האחרון במערך

        lastDiminisher(Agent[]agents\agents[n-1]);

}
```

<u>להוכיח שמנגנון יעיל פארטו:</u>

להניח בשלילה שקיים שיפור ולהגיע לסתירה.

<u>להוכיח שמנגנון אינו יעיל פארטו:</u>

דוגמא נגדית שמתקבלת מהמנגנון אך קיים לה שיפור.

<u>להוכיח שמנגנון הוא ללא קנאה:</u>

להוכיח מקסום סכום הערכים. הבחירה של המנגנון היא הערך המקסימלי עבור כל חלק/פריט

<u>להוכיח שמנגנון הוא עם קנאה:</u>

דוגמה של מצב המתקבל מהמנגנון בו יש לפחות אחד מהסוכנים שמעוניין להחליף את חלקו.

```
bool PerfectMatching (graph G){…} //פונקציה הבודקת אם קיים שידוך מושלם

MarkBestRooms(Agent a, graph G){  //פונקציה יוצרת עבור הסוכן קשתות אל חדרים מועדפים
  int bestValue=0; //אחסון הערך הגבוה ביותר
  for(i=0 to n){    //מעבר על כל החדרים
    int value = a.eval(i); //קביעת הערך של החדר הספציפי
      if(value > bestValue) {
        bestValue =value;   // שכתוב ההצעה הכי טובה
        G.removeEdge(a-> all edges); //נקה הצעות קודמות
        G.addEdge(from a-> room i); //סמן הצעה נוכחית
      }
      if(value=bestOffer) G.addEdge(from agent to room j)//סמן הצעה
  }
}

findEnvyFreeAssignment(Agent[]agents){
  graph G (n×n);  //גרף ההצעות הטובות ביותר
  for(i=0 to n){    //מעבר על כל הסוכנים
     MarkBestRooms(agents[i],G);  //סימון החדר/ים המועדף ביותר על הסוכן
  }

  If (PerfectMatching(G)) Print the graph;  //אם קיים שידוך מושלם, הדפס אותו
  Else{
    print("An envy-free assignment is not exist");
  }
}
```