Задача

- Дадени са $\boldsymbol{n}$ изчислителни машини с еднаква скорост и $\boldsymbol{m}$ задачи, които могат да бъдат изпълнени ($\boldsymbol{n} < \boldsymbol{m}$). Всяка задача е дадена с време за изпълнение $\boldsymbol{t_i}$ и печалба $\boldsymbol{p_i}$, ако бъде изпълнена;

- Да се състави евристичен алгоритъм (базиран на изкачване с рестартиране, табу търсене или симулирано закаляване) намиращ, кои задачи да се изпълнят, за да бъде печалбата максимална, ако машините могат да бъдат наети за време **T**.

```
public static int Evaluate(
    int[] t,
    int[] p,
    int T,
    int[] solution)
{
    var result = 0;
    for (int i = 0; i < solution.Length; i++)
    {
        if(solution[i] == 1)
        {
            T -= t[i];
            if (T < 0)
                return -1;
            else
                result += p[i];
        }
    }

    return result;
}
```

```
public static int[] RandomSolution(
    Random random,
    int size)
{
    var result = new int[size];
    for (int i = 0; i < size; i++)
        result[i] = random.Next(2);

    return result;
}
```

```
public static int[] NextSolution(
    int[] t,
    int[] p,
    int T,
    int[] solution)
{
    var result = solution
        .ToArray();

    var bestIdx = -1;
    var bestVal = int.MinValue;
    for (int i = 0; i < result.Length; i++)
    {
        result[i] = Math.Abs(result[i] - 1);
        var val = Evaluate(t, p, T, result);
        if (val > bestVal)
        {
            bestVal = val;
            bestIdx = i;
        }
        result[i] = Math.Abs(result[i] - 1);
    }

    result[bestIdx] = Math.Abs(result[bestIdx] - 1);
    return result;
}
```

```
public static int[] HCSARR(
    Random random,
    int[] t,
    int[] p,
    int T,
    DateTime deadLine)
{
    var bestVal = int.MinValue;
    var bestSol = RandomSolution(random, t.Length);
    var currVal = bestVal;
    var currSol = bestSol;
    while (DateTime.Now < deadLine)
    {
        var newSol = NextSolution(t, p, T, currSol);
        var newVal = Evaluate(t, p, T, newSol);
        if (newVal > currVal)
        {
            currVal = newVal;
            currSol = newSol;
        }
        else
        {
            currSol = RandomSolution(random, t.Length);
            currVal = Evaluate(t, p, T, currSol);
        }

        if (currVal > bestVal)
        {
            bestVal = currVal;
            bestSol = currSol;
        }
    }
    return bestSol;
}
```