„Algoritmų sudarymas ir analizė“ (P170B400) Inžinerinio projekto Nr.2 ataskaita
Parengė IFF-5/4 gr. Studentas Mantas Damijonaitis

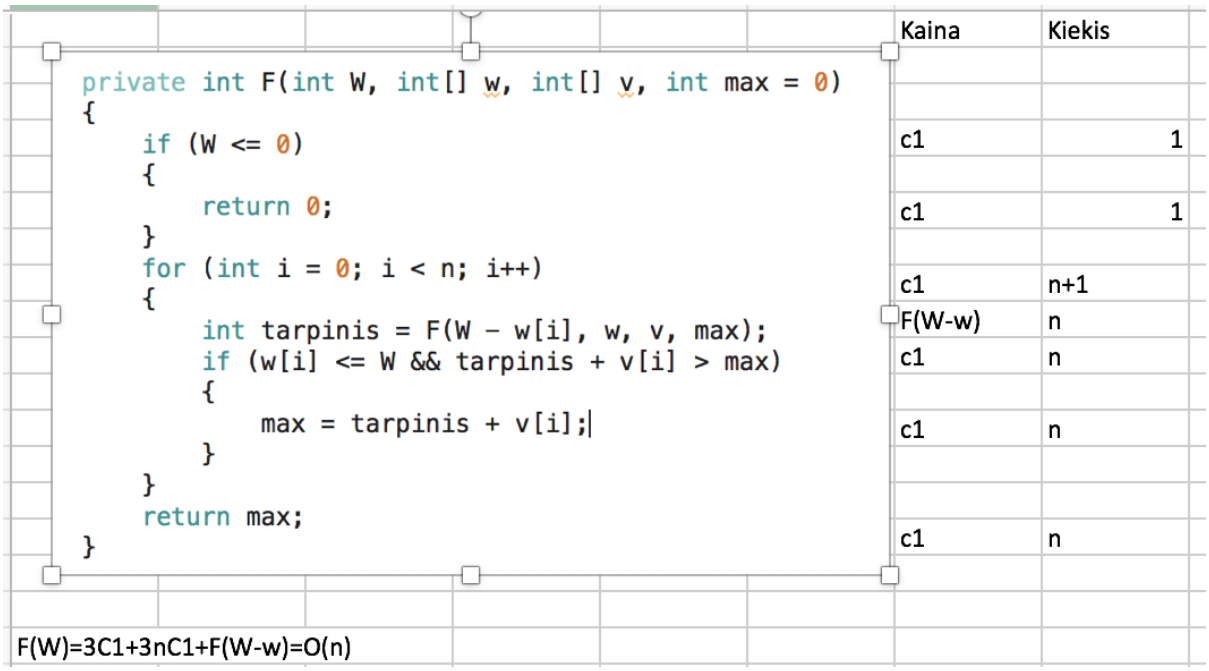
1. Uždavinys: duotą rekurentinę lygtį išpręsti naudojant rekursiją, bei dinaminio programavimo metodus.

   Duota: n, W, $v_1, v_2, v_3, \dots, v_n$ ir $w_1, w_2, w_3, \dots, w_n$
   $F(W) = \max_{i:w <= w} \{ F(W - w_i) + v_i \}$, F(0)=0

   Metodo realizacija panaudojant rekursiją:

```
private int F(int W, int[] w, int[] v, int max = 0)
{
    if (W <= 0)
    {
        return 0;
    }
    for (int i = 0; i < n; i++)
    {
        int tarpinis = F(W - w[i], w, v, max);
        if (w[i] <= W && tarpinis + v[i] > max)
        {
            max = tarpinis + v[i];
        }
    }
    return max;
}
```

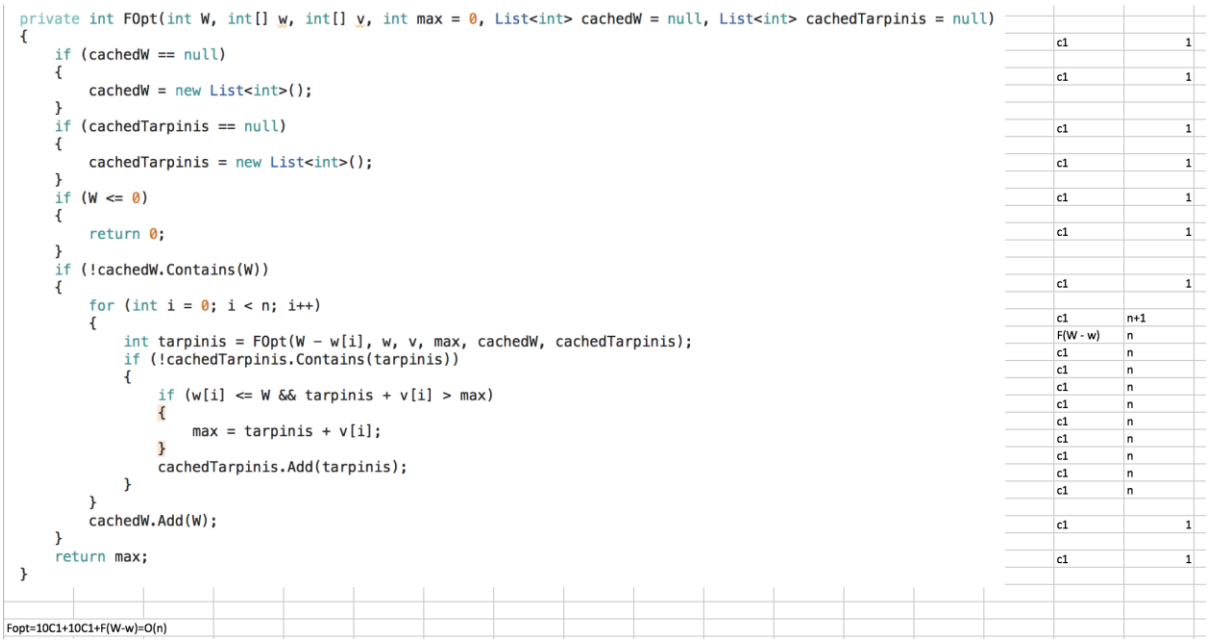
   Metodo sudėtingumas:



Metodo realizacija išsisaugant tarpinius rezultatus (panaudojant dinaminio programavimo principus):

```
private int FOpt(int W, int[] w, int[] v, int max = 0, List<int> cachedW = null, List<int> cachedTarpinis = null)
{
    if (cachedW == null)
    {
        cachedW = new List<int>();
    }
    if (cachedTarpinis == null)
    {
        cachedTarpinis = new List<int>();
    }
    if (W <= 0)
    {
        return 0;
    }
    if (!cachedW.Contains(W))
    {
        for (int i = 0; i < n; i++)
        {
            int tarpinis = FOpt(W - w[i], w, v, max, cachedW, cachedTarpinis);
            if (!cachedTarpinis.Contains(tarpinis))
            {
                if (w[i] <= W && tarpinis + v[i] > max)
                {
                    max = tarpinis + v[i];
                }
                cachedTarpinis.Add(tarpinis);
            }
        }
        cachedW.Add(W);
    }
    return max;
}
```

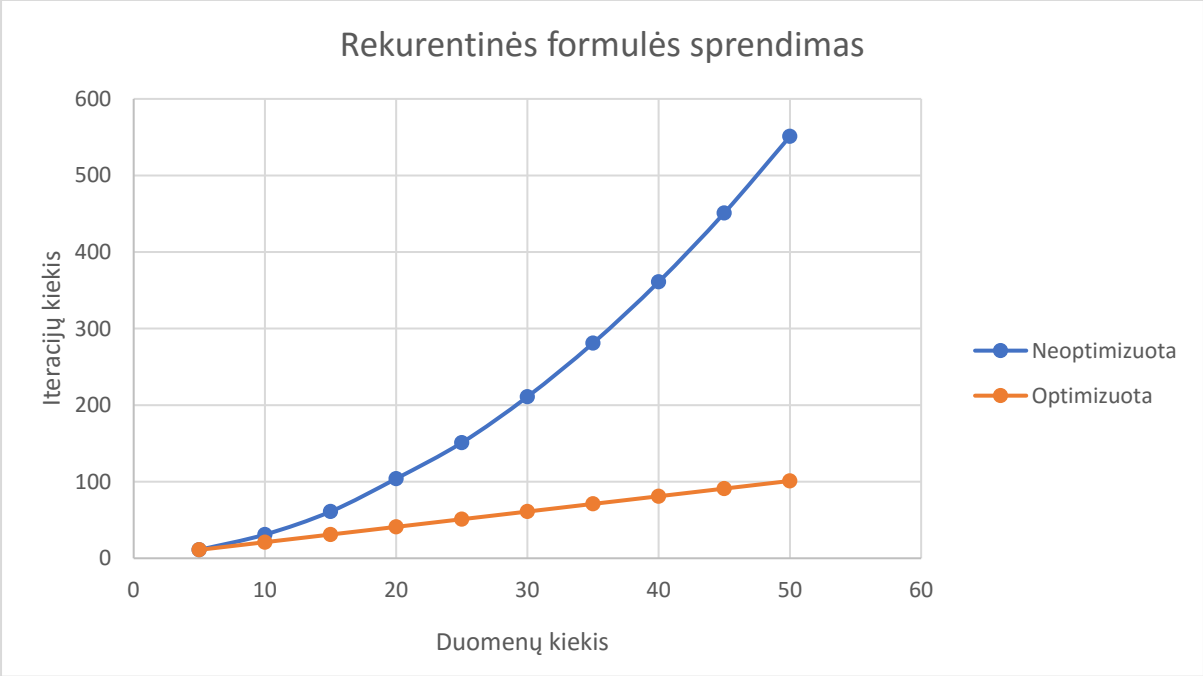
Metodo sudėtingumas:



Laboratoriniai bandymai:

Pradiniai duomenys	w	v	W
	1	1	2
	1	1	
	2	2	
	7	7	
	5	5	
	6	6	
	7	7	
	7	7	
	7	7	
	4	4	
Rezultatas: 3			
Neoptimizavus funkcija pakviesta kartu		30	
Optimizavus funkcija kviesta kartu:		21	

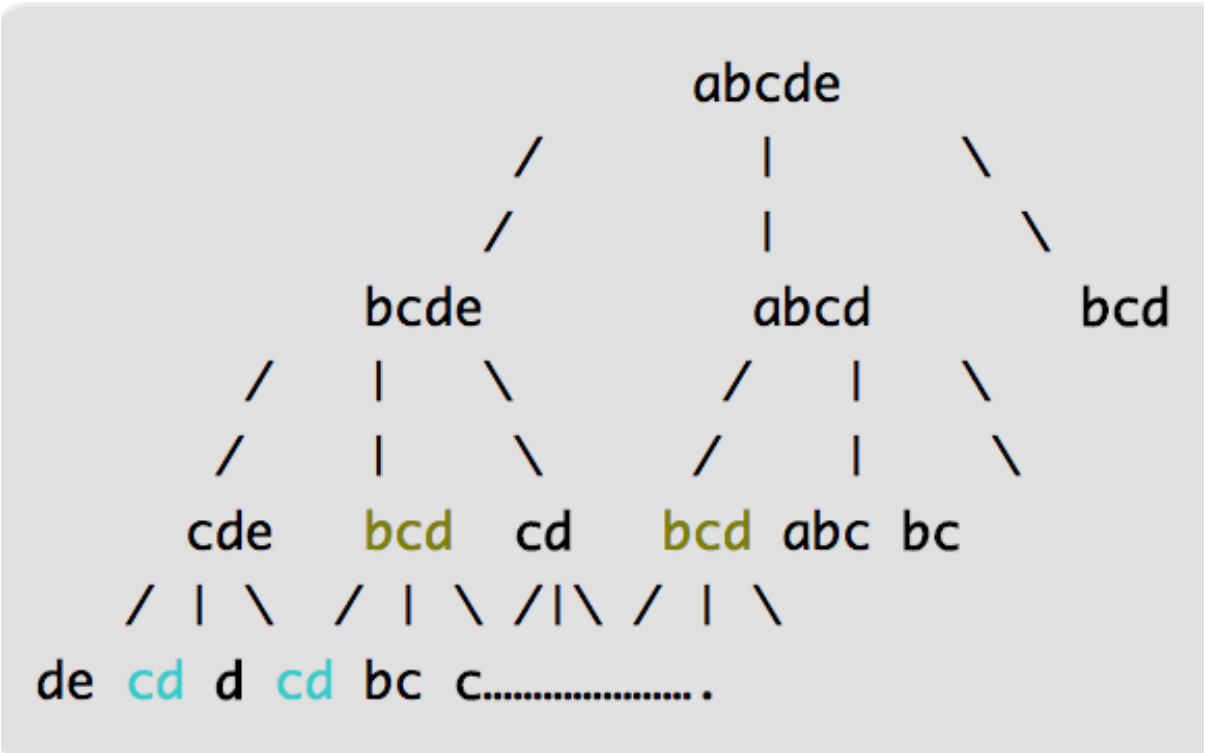
Pradiniai duomenys	w	v	W
	1	1	2
	1	1	
	1	1	
	1	1	
	5	5	
	6	6	
	7	7	
	8	8	
	9	9	
	10	10	
Rezultatas: 5			
Neoptimizavus funkcija pakviesta kartu		51	
Optimizavus funkcija kviesta kartu:		21	



2. Uždavinys:

Duota simbolių seka. Raskite mažiausią kiekį simbolių kuriuos reiktų įterpti, kad seka taptų simetrinė.

Optimizacijos idėja: analizuodami problemos sprendimą, suvokiame, kad tenka spręsti panašaus tipo problemas, skaldant jas į vis mažesnes dalis:



Šiuo atveju cd – pasikartojanti problema, kuri sprendžiant rekursiškai, bereikalingai nagrinėjama daug kartų.

Sprendimas: išskaidžius sprendimą į daug dalių jau gautus sprendimus laikyti lentelėje, kad juos būtų galima panaudoti dar kartą.

```
a b c d e
---------
0 1 2 3 4
0 0 1 2 3
0 0 0 1 2
0 0 0 0 1
0 0 0 0 0
```

Metodo realizacija:

```
private static int MinInsertionsToFormPalyndrome(string word)
{
    int[,] minInsertionsTable = new int[word.Length, word.Length];
    for (int gap = 1; gap < word.Length; gap++)
    {
        for (int l = 0, h = gap; h < word.Length ; l++, h++)
        {
            minInsertionsTable[l, h] = word[l] == word[h] ?
                minInsertionsTable[l + 1, h - 1] :
                GetLowerNumber(minInsertionsTable[l, h - 1], minInsertionsTable[l + 1, h]) + 1;
        }
    }
    return minInsertionsTable[0, word.Length - 1];
}

private static int GetLowerNumber(int a, int b)
{
    return a < b ? a : b;
}
```

Algoritmo sudėtingumas: O (n^2) (šaltinis - [http://www.geeksforgeeks.org/dynamic-programming-set-28-minimum-insertions-to-form-a-palindrome/)](http://www.geeksforgeeks.org/dynamic-programming-set-28-minimum-insertions-to-form-a-palindrome/)

Testiniai rezultatai bei programos atsakymai:

“geeks” – 3 (skgeeg**ks)**
“Ab3bd” – 2 (Adb3bdA)

Laboratoriniai bandymai: