6 4 3

Iščemo takšno izbiro strank, da bo vsota izbranih števil (tj. mest v parlamentu) večja kot vsota neizbranih, ampak med takimi izbiramo najmanjša.

S – tisti, ki smo jih izbrali

N – tisti, ki jih nismo

Zahtevamo $(\text{vsota } S) > \dfrac{\text{vsota vseh}}{2} = \dfrac{(\text{vsota } S) + (\text{vsota } N)}{2}$

Če iz obeh strani odštejem $\dfrac{(\text{vsota } S)}{2}$, dobim

$$\frac{1}{2}(\text{vsota } S) > \frac{1}{2}(\text{vsota } N)$$

torej $(\text{vsota } S) > (\text{vsota } N)$

Očitna rešitev: Poizkusimo vse možnosti

4 4 3 8 10 9 5

Poizkusimo vse možnosti:

16	27	~~4 4 3 8 10 9 5~~	~~4 4 3 8 10 9 5~~	4	39
18	25	~~4 4 3 8 10 9 5~~	4 4 3 8 10 9 5	35	8
16	27	~~4 4 3 8 10 9 5~~	4 4 3 8 10 9 5	43	0
20	23	~~4 4 3 8 10 9 5~~	~~4 4 3 8 10 9 5~~	0	43

Od izborov, ki so tu zapisani, je izbor

3, 8, 10, 9, 5 optimalen

tj. izmed izborov, pri katerih je $(\text{vsota } S) > (\text{vsota } N)$

ima ta izbor najmanjšo (vsota S)

Recimo, da bi želeli na roke pregledati <u>vse</u> možne izbore.
Koliko dela bi imeli s tem?

Če imamo n števil (v našem primeru je $n=7$),
potem imamo $\boxed{2^n}$ možnih izborov

V našem primeru $2^7 = 128$

Dodaten problem: Pri takih eksponentnih rasteh postanejo številke hitro ZELO velike.

Npr. če je $n=30$, je $2^n \approx$ milijarda

Ker je $n \leq 20$, in $2^{20} \approx$ milijon, si lahko privoščimo, da bi pregledali vse možnosti!

Če bi bil $n \leq 30$, pa tega ne bi mogli.

SKLEP: Časovno gledano bo naša rešitev dovolj hitra glede na te omejitve.

Predlagana optimizacija: Namesto da gledamo vseh 2^n možnosti, jih pregledamo le 2^{n-1} in vzamemo večji del kot izhod.

4	4	3	8	10	9	5	|	4	4	3	8	10	9	5
		↑			↑	↑				↑			↑	↑

26 17 17 26

Še vedno imamo eno težavo: Kako dejansko preverimo vse možnosti?

Takemu pristopu k reševanju problemov pravimo
Complete search (brute force).

Lahko probamo tako:

Recimo, da smo se vnaprej odločili, da bomo vzeli točno k števil. ZNAMO!

Če je $k=1$, lahko to naredimo z eno zanko:

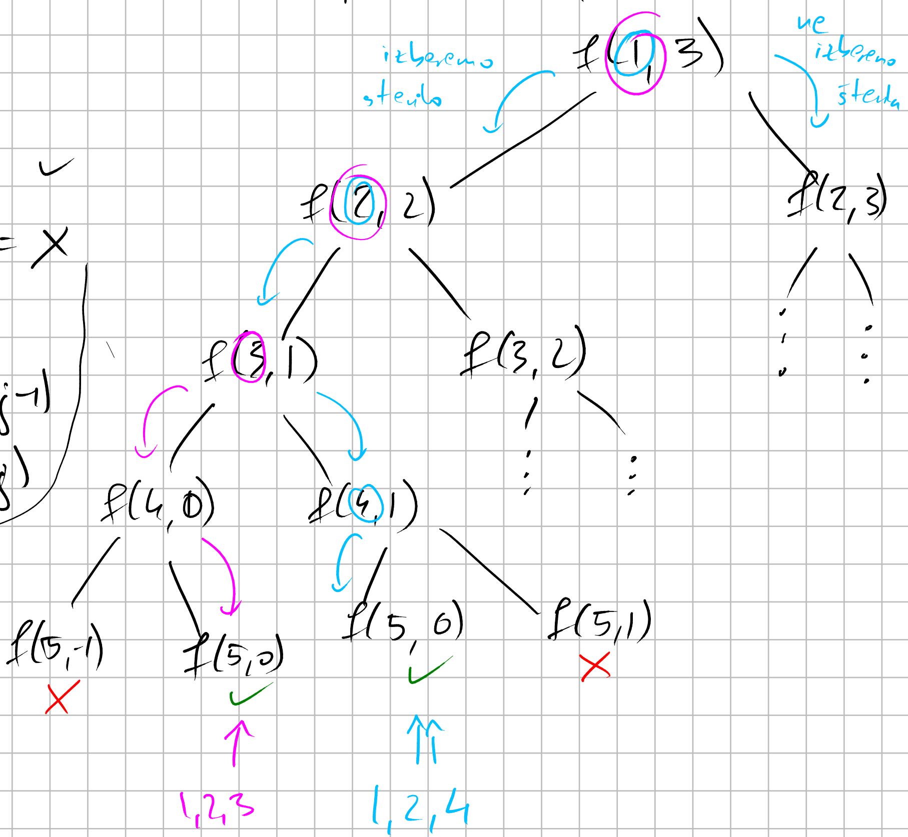
```
for (int i=0; i<n; i++) {
    // izberemo i-ti element in le tega
}
```

Če je $k=2$, lahko to naredimo z dvema zankama:

```
for (int i=0; i<n; i++){
    for (int j=0; j<n; j++){
        if (i==j) continue;
        // izberemo elementa i in j
    }
}
```

Kaj pa če ne poznamo (vnaprej) števila k?
Ne moremo vnaprej napisati neznanega števila zank!
Pravo orodje je rekurzija!

Primer: $n=4$, $k=3$



Na kratko:

$f(n+1, 0) = \checkmark$

$f(n+1, \neq 0) = \times$

$f(i, j) =$
$\quad f(i+1, j-1)$
$\quad f(i+1, j)$

Kako pa poiščemo vse možne izbore?

Lahko gremo z eno zanko skozi vse možne k in

pokličemo vsakič znova $f(1, k)$

Rešili smo problem!