```
a)
    dolžina (dm) 1  2   3   4   5   6   7   8   9
    cena (EUR)   2  4   7   9   10  11  12  19  22
    dobiček (ri) 2  4   7   9   11  14  16  19  22
    pozicija rez 1  2   3   4   1   3   1   8   9

    7: splača se jim razrezati palico 6 in vzeti palico 1 -> skupen dobiček = 16€
    8: splača se jim kar prodati palico samo (dolžine 8)  -> skupni dobiček = 19€
    9: splača se jim kar prodati palico samo (dolžine 9)  -> skupni dobiček = 22€

b)
    V PRILOGI
c)
    najbolje da si zapomnimo rešitve od najmanjše dolžine proti največje kolikor gre
    saj se te dolžine v rekurziji največkrat kličejo
    torej si zapomnimo prvih K elementov in njihovih dobičkov

    ko primerjam v kodi:
        prvi = maxPrice
        drugi = value[j] + solution[i - (j+1)]
    tu bi rabil preveriti ali solution[i - (j+1)] obstaja. če ne - se požene
    rekurzija nad to dolžino


    zdaj je časovna kompleksnost O(n^2), pri tem primeru se n-k krat kliče
    rekurzija, ki pa potuje v globino 2^(n-k) dolžine, saj je K elementov že izračunanih

    tako da je časovna kompleksnot -> n^2 * 2^(n-k)

    večji kot izberemo K, bolj se približamo n^2, za vsak večji K je treba izračunati
    potenco števila 2 manj primerov saj jih imamo že shranjenih
```

```python
DICT = {  # dolzina ->[cena, dobicek]
    1: [2, 2],
    2: [4, 4],
    3: [7, 7],
    4: [9, 9],
    5: [10, 10],
    6: [11, 11],
    7: [12, 12],
    8: [19, 19],
    9: [22, 22]
}
def calcProfit(minus=0):
    value = [x1[0] for x1 in DICT.values()]
    len = max(DICT.keys())
    solution = [0] * (int(len) + 1)
    for i in range(len + 1):
        maxPrice = value[i - 1]
        for j in range(i):
            prvi = maxPrice
            drugi = value[j] + solution[i - (j + 1)]
            ###### SPODAJ SO TRIJE PRIMERI --- ZA OBIČAJNO -> MINUS ==0, MINUS -> MINUS >0 in RAZMERJE -> BA
            if minus < 0:  ##BA
                razmerje = value[j] if value[j] > solution[i - (j + 1)] else solution[i - (j + 1)]  ##RAZMERJE B/AŽ
                if prvi <= drugi + razmerje:
                    maxPrice = prvi
                else:
                    maxPrice = drugi
            elif minus > 0:  # tukaj odšteje -2
                if prvi <= drugi + minus:
                    maxPrice = prvi
                else:
                    maxPrice = drugi
            else:  ##minus == 0
                maxPrice = max(prvi, drugi)
            solution[i] = maxPrice
            DICT[i][1] = maxPrice
    i = 1
    for key, arr in DICT.items():
        print(str(key) + " " + str(arr))
        i += 1
def main():
    # 1. - 2€ pri rezu
    calcProfit(2)  ## ODŠTEJE -2 PRI REZU
    print()  ##emty line
    # 2. -(b/a)€  -- b/a je razmerje rezov kjer je a krajša palica
    calcProfit(-1)  ##ODŠTEJE RAZMERJE B/A PRI REZU
main()
```

rezultati -2

```
1 [2, 2]
2 [4, 4]
3 [7, 7]
4 [9, 9]
5 [10, 10]
6 [11, 11]
7 [12, 12]
8 [19, 14]
9 [22, 16]
```

rezultati b/a

```
1 [2, 2]
2 [4, 4]
3 [7, 7]
4 [9, 9]
5 [10, 10]
6 [11, 11]
7 [12, 12]
8 [19, 19]
9 [22, 22]
```