1. Slozenost izracunavanja
   - najcesce je korektnost programa primarni zahtev
   - nakon toga su bitne performanse programa
   - za svaki program postoji vise razlicitih resenja koja mogu biti vise ili manje efikasna
2. Racunar ima procesor, radnu memoriju RAM i trajno skladise – hard disk
   - u RAM-u se cuvaju programi dok se izvrsavaju (tu se nalaze data segment, hip segment i segment koda)
   - operativni sistem brise program iz RAM memorije (a na hard disku ostaje program) kada se program zavrsi
   - RAM memorija je brza od hard diska i program ne koristi hard disk za ovaj program koji se izvrsava; ali sporija od samog procesora CPU
   - tako da procesor moze da izvrsava vise programa od jednom
   - imamo Kes nivoa 1 – najbrza i najmanja memorija, kes nivoa 2 – malo manje brza i malo veca… kes nivoa 3… i ti nivoi redom su povezani sa radnom memorijom
   - kes pamti podatke iz rama u kesu, ako se ne nalazi u k1, trazi u k2 …
   - ram cesto vraca ceo blok vrednosti (linija kesa, kes line) i kes pamti ceo taj blok memorije
     - to je korisno ako imamo niz u programu i ako trazimo prvi element niza, deo niza ce biti kopiran i to stedi vreme i ne moramo da skacemo do ram memorije (mozda cak i ceo niz) – ovo je ideja sekvencijalnog pristupa memoriji i moze uticati na vreme izvrsavanja programa
     - razliciti nivoi optimizacije se mogu navoditi prilikom prevodjenja 0,1,2,3,s
     - nivo 0 - osnovne optimizacije, skoro zanemarljive
     - nivo 1 – kompilator pokusava da smanji velicinu programa i vreme izvrsavanja programa (bez usporavanja kompilacije, jeftine optimizacije)
     - nivo 2 – optimizacije koje ubrzavaju program ali bez zauzeca dodatne memorije (mogu da zahtevaju vise vremena tokom kompliacije programa)
     - nivo 3 – optimizacije koje ubrzavaju program i smeju da koriste dodatna polja u ramu (ukljucuje sve prethodne optimizacije plus…)
     - nivo s – optimizacija velicine izvrsnog fajla, ne uzimajuci u obzir velicinu memorije i brzinu izvrsavanja (kao nivo 3 bez optimizacija koje povecavaju velicinu izvrsnog fajla)
       - najcesce se koristi nivo 2 ili 3, kada se koriste mali uredjaji i mali cipovi onda se koristi s (aparati u kuhinji)
3. Kompajler optimizuje:
   - logicki &, |, << , >> su brze
   - binarni operatori su brze operacije +, -, &&, ||
   - mnozenje * je u procesoru komplikovano implementirana i bice sporija, /, % isto malo sporije zbog njihove implementacije
     - 2*i se moze brze dobiti ako se napise i<<1
     - i/4 se brze dobija i>>2
     - kompajleri ovo znaju da optimizuju i jos mnogo toga (tako da su cesto i bolji od rucne optimizacije koju programer uradi)
     - ovo su trivijalne optimizacije i spadaju u nivo 1
     - nivo 2 i 3 su komplikovanije
   -
4. Slozenost algoritma
   - odnosi se na matematicku procenu broja operacija u nasem kodu
     - procena zavisi od parametra naseg programa, vrednosti n
     - n moze da bude dimenzija niza
     - n moze da bude broj koji se unosi sa standardnog ulaza
     - n moze da bude broj bitova u zapisu broja

    -
    - nema veze sa prethodnim pojmovima
    - govorimo najcesce o broju operacija u najgorem mogucem slucaju, maksimalan moguci broj operacija
    - od toga zavisi vreme izvrsavanja naseg programa
    - druga dimenzija je … ILI o velicini memorije koja se zauzima u toku izvrsavanja programa
    - u zavisnosti od toga govorimo o VREMENSKOJ SLOZENOSTI I MEMORIJSKOJ slozenosti
        - ako imamo pretrazivanje celog niza od pocetka do kraja, vremenska slozenost je u najgorem slucaju n poredjenja, a memorijsku slozenost smatramo konstantnom jer koristimo samo jednu dodatnu promenljivu
        - ako zelimo da sortiramo ceo niz, to nikako ne moze da se uradi u n koraka, znatno komplikovanija operacija
    -
    -
    - …
5. Kada govorimo o (vremenskoj ili memorijskoj) slozenosti algoritma, podrazumevamo situaciju kada dimenzija problema raste
    - u ovoj situaciji ne posmatramo male vrednosti dimenzije problema, vec situaciju koja se desava sa povecanjem dimenzije problema
    - SLIKA IZ KNJIGE ILI AKO JE IMA NA SLAJDOVIMA
6. SLAJD 16, DEFINICIJA VELIKO O
    - klasa fja, skupovna pripadnost, ne prava jednakost
    - SLAJD 21, SLICICE
    - g ogranicava fju f za velike vrednosti
7. STRANA 43 – KNJIGA P2 - OZNAKA OMEGA – ogranicenje sa donje strane
8. OZNAKA TETA – ogranicava i sa gornje i sa donje strane
    - cesce se koristi u programiranju ogranicenje veliko O a ne veliko teta
    - na taj nacin je obradjen najgori slucaj, ali uzima se minimalno g tako da vazi f = O(g)
    -