

Algoritmin toimintaidea pseudokoodina:

```

function TOPOALGORITMI(G)
  pino = Topological-Sort(G)
  while pino NOT EMPTY do
    u = pino.pop()
    for all v in vierus[u] do
      RELAX(u,v)
    end for
  end while
end function

function RELAX(u,v)
  if pisteSaalis[v] < pisteSaalis[u] + v.pisteArvo then
    pisteSaalis[v] = pisteSaalis[u] + v.pisteArvo
    reitti[v] = u
  end if
end function

```

Algoritmi kutsuu ensin verkolle G Topological-Sort(G) -algoritmia, jonka aikavaativuus on tunnetusti $O(|V|+|E|)$, missä V on solmujen ja E kaarien joukko. Sitten While-looppi käy läpi kerran jokaisen solmun verkossa eli suorittaa itsensä |V| kertaa. Relax-metodi on aikavaativuudeltaan vakio. Koko algoritmin aikavaativuudeksi tulee siis $O(2|V|+|E|)=O(|V|+|E|)$. Käytännössä algoritmi vielä lopuksi etsii parhaan pistemäärän taulukosta pisteSaalis ja tulkitsee parhaan reitin taulukosta reitti. Nämä toiminnot ovat kuitenkin lineaarisia solmujen määrän suhteen, joten aikavaativuusarvio säilyy samana.

Algoritmi käyttää aputaulukkoja pisteSaalis, reitti sekä pinoa. Lisäksi Topological-Sort -algoritmi käyttää aputaulukkoa kayty. Kaikkien taulukkojen ja pinon tilavaativuus on $O(|V|)$. Koko algoritmin tilavaativuus on siis $O(4|V|)=O(|V|)$.

Algoritmin eräs puutos on, että se ei tunnista, jos sille annetaan verkko, jossa on sykli. Algoritmi ei siis toimi oikein, jos verkossa on sykli. Käytännössä ongelmia tulee, kun algoritmi yrittää tulkita reitti-taulukkoa ja algoritmi jää looppiin.