$$t(a,a) = 0$$

$$t(a,c) = \max_{<a,b>\in E} (eval(a,b) + t(b,c))$$

```
public double tExp(Vertex a, Vertex c, DiGraph g) {
   if (a == c) return 0 ;
   else {
         Vertex b ;
         double max = 0, temp;
         Edge e = g.firstEdge(a) ;
         while (e != null) {
             b = g.endPoint(e) ;
             temp = ((Double)e.evalue).doubleValue() + tExp(b,c, g);
                                                          // rekurzija
             if (temp > max)
                   max = temp ;
             e = g.nextEdge(a, e) ;
         } // while
         return max ;
      } // else
   } // tExp
```

KRITIČNA POT – DINAMIČNO PROGRAMIRANJE

INICIALIZACIJA:
- izračunamo vstopne stopnje vseh vozlišč
- postavimo začetne čase za vsa vozlišča na 0

```
class ValueType {
    String name ;
    int inDegree;
// Vertex parent ; // kazalec na predhodnika
    double time ;
  } // class ValueType
```

Kritična pot – dinamično programiranje

```
public double tDynamic(Vertex a, Vertex c, DiGraph g) {
   // a – zacetno vozlisce, c – zakljucno vozlisce
   Vertex v, w ; // trenutno vozlisce in njegov naslednik
   Edge e ; // povezava <v, w>
   List ls = new ListLinked(); // seznam vozlisc, katerih
                               // naslednikov se nismo pregledali
   Object pos ;
```

```
ls.insert(a);
while (! ls.empty()) {
    pos = ls.first() ; // izberemo lahko poljubno vozlisce
    v = (Vertex)ls.retrieve(pos) ; // izberemo kar prvega
    ls.delete(pos);
    e = g.firstEdge(v);
    while (e != null) {
        w = g.endPoint(e);
        if (((ValueType)w.value).time < ((ValueType)v.value).time +
                                        ((Double)e.evalue).doubleValue())
          ( (ValueType) w.value).time = ((ValueType)v.value).time +
                                        ((Double)e.evalue).doubleValue();
       ((ValueType)w.value).inDegree -- ;//ena pot do w pregledana
        //ce so pregledane vse poti do w,je w kandidat za pregledovanje
        if ( ((ValueType)w.value).inDegree == 0)
            ls.insert(w);
        e = g.nextEdge(v, e) ;
      } // while e != null
    } // while ! ls.empty()
    // koncni cas je cas zakljucnega vozlisca
    return ((ValueType)c.value).time ;
} // tDynamic
```

Shranitev in izpis kritične poti

```
if (((ValueType)w.value).time < ((ValueType)v.value).time +
                                ((Double)e.evalue).doubleValue()) {
   ( (ValueType) w.value).time = ((ValueType)v.value).time +
                                ((Double)e.evalue).doubleValue();
   ( (ValueType) w.value).parent = v; // *** dodano ***
}
```

```
// izpis kriticne poti
w = c ;
while (w != a) {
   v = ((ValueType)w.value).parent ;
   e = g.firstEdge(v) ;
   while (g.endPoint(e) != w)
      e = g.nextEdge(v, e) ;
   System.out.println("<" + v + ", " + w + ", " + e + ">") ;
   w = v ;
} // while
```