IMPLEMENTASI ALGORITMA RUNUT BALIK DALAM MENYELESAIKAN PERMAINAN *LOGIC BRIDGE*

Putri Erivani – NIM 13505033

Program Studi Teknik Informatika,Sekolah Teknik Elektro dan Informatika,
Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if115033@students.if.itb.ac.id

ABSTRAK

Banyak algoritma yang dapat diimplementasikan untuk menyelesaikan berbagai permainan logika (*mind games*). Salah satunya adalah algoritma runut balik (*backtracking*). Algoritma runut balik pertama kali diperkenalkan pada tahun 1950 oleh D.H. Lehner. Algoritma ini banyak diimplementasikan dalam program permainan (misalnya *tic-tac-toe*, *maze*, sudoku, catur, dll) dan dalam bidang *artificial intelligence.*

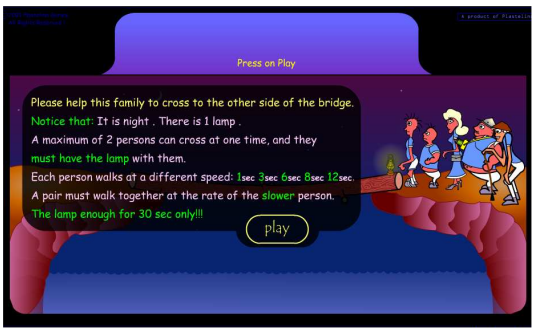
Makalah ini membahas implementasi algoritma runut balik untuk menyelesaikan permainan *logic bridge*, baik algoritma penyelesaian, pembentukan pohon secara dinamis, maupun kompleksitas waktu algoritmanya.

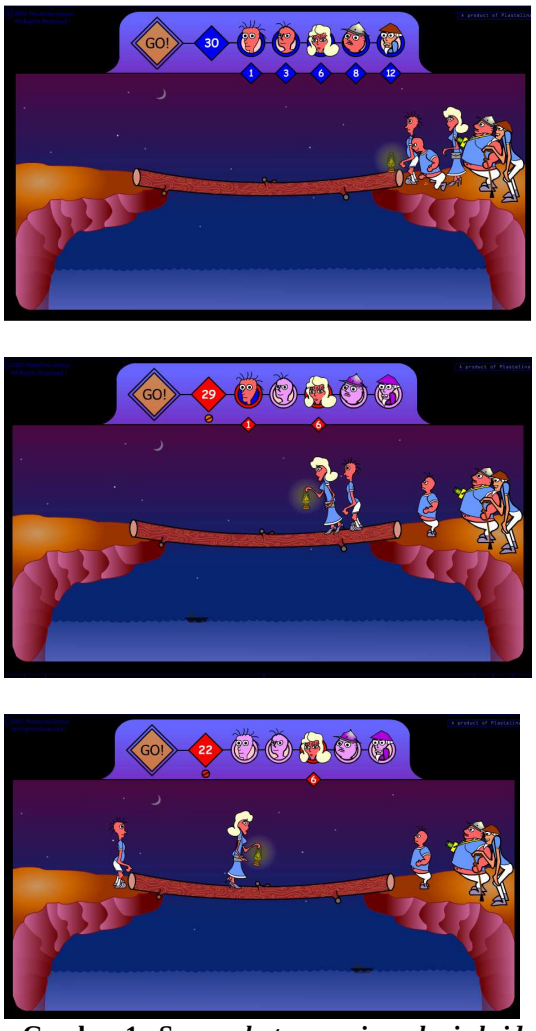
Kata kunci : *backtracking*, *logic bridge*, pohon dinamis, kompleksitas algoritma.

1. PERMAINAN LOGIC BRIDGE

Permainan *logic bridge* adalah permainan dimana pemainnya diminta menolong sebuah keluarga menyeberangi jembatan di malam hari. Keluarga tersebut terdiri dari lima orang dan hanya memiliki satu lampu. Jumlah maksimal orang yang menyeberang pada suatu waktu adalah dua orang, dan ketika menyeberang mereka harus membawa lampu. Setiap orang memiliki waktu menyeberang yang berbeda-beda, yaitu 1 detik, 3 detik, 6 detik, 8 detik, dan 12 detik. Pasangan yang menyeberang harus berjalan bersama-sama dengan kecepatan orang yang lebih lambat, sedangkan lampu hanya dapat bertahan selama 30 detik.

Screenshot permainan ini dapat dilihat pada gambar 1.





Gambar 1. *Screenshot* permainan *logic bridge*

2. PENYELESAIAN PERMAINAN *LOGIC BRIDGE* DENGAN ALGORITMA RUNUT BALIK

Penyelesaian permainan *logic bridge* ini memiliki pohon ruang status yang sangat besar, yaitu memiliki daun sebanyak C(5,2) x C(2,1) x C(4,2) x C(3,1)) x C(3,2) x C(4,1)) x C(2,2) = 5760 buah. Sebagian pohon ruang status statis permainan *logic bridge* dapat dilihat pada gambar 2.

Untuk mereduksi pembangkitan simpul, pencarian solusi permainan dilakukan dengan pembentukan pohon secara dinamis. Langkah-langkah pencarian solusi dengan algoritma runut balik adalah sebagai berikut [RM07]:

1. solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai adalah metode pencarian mendalam (DFS). Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup. Simpul hidup yang sedang diperluas dinamakan simpul-E. Simpul diberi nomor sesuai urutan kelahirannya.
2. Tiap kali simpul diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang dibentuknya tidak menuju ke solusi, maka simpul-E tersebut menjadi simpul mati dan tidak akan diperluas lagi. Fungsi yang membunuh simpul-E adalah fungsi pembatas.
3. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian diteruskan dengan membangkitkan simpul anak lainnya. Bila tidak ada lagi simpul anak yang dapat dibangkitkan, maka pencarian solusi dilanjutkan dengan melakukan *backtrack* ke simpul orang tuanya. Selanjutnya simpul ini menjadi simpul hidup yang baru. Lintasan baru dibangun kembali sampai lintasan tersebut membentuk solusi.
4. pencarian dihentikan bila kita telah menemukan solusi atau tidak ada lagi simpul hidup untuk runut balik.

Tinjau persoalan *logic bridge* dengan instansiasi :

$N = 5$

$(t_1,t_2,t_3,t_4,t_5) = (1,3,6,8,12)$

$T = 30$

Solusi dinyatakan sebagai $X = ((x_1, x_2), x_3,(x_4, x_5), x_6, (x_7, x_8), x_9, (x_{10}, x_{11})), x_i \in \{1,3,6,8,12\}$. Fungsi pembatas yang digunakan adalah

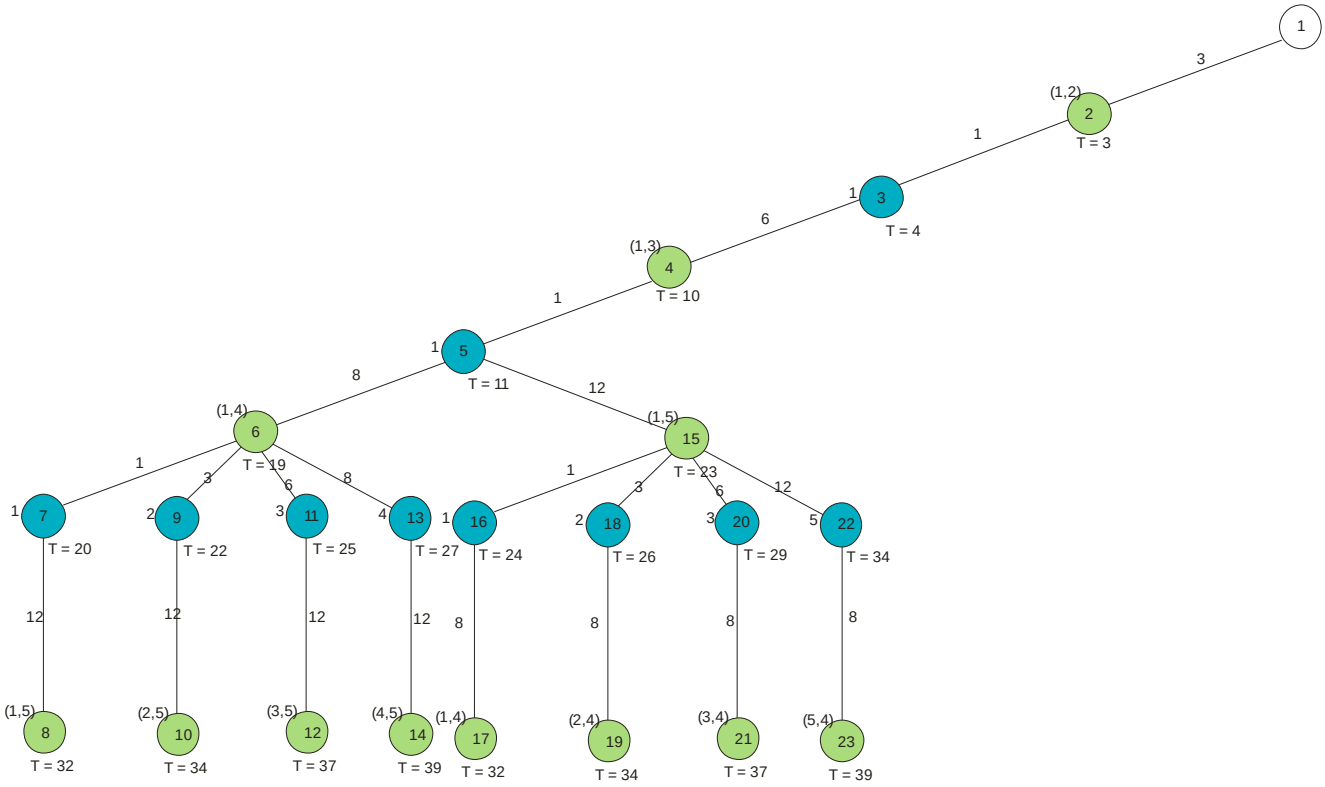
$$\sum_{k=1}^{n} xk \le M$$

Dengan nilai x_k dari (x_i,x_j) adalah nilai terbesar dari x_i,x_j.

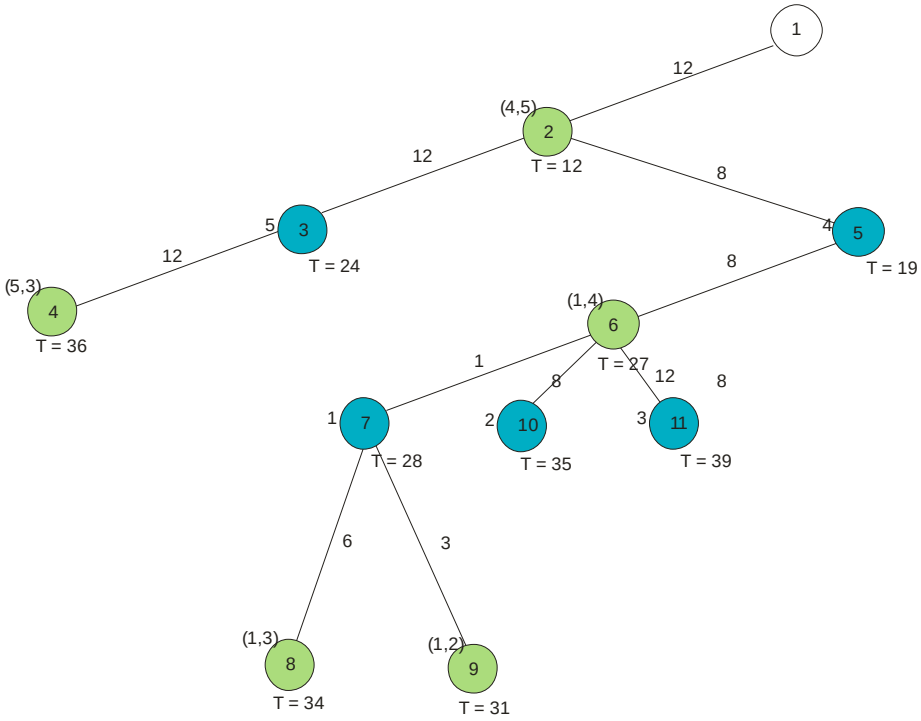
Jadi, jika vektor solusi tidak memenuhi ketidaksamaan tersebut, maka lintasan di dalam pohon pencarian lintasan tersebut tidak diperpanjang lagi.

Pencarian dimulai dari status awal (simpul akar) dimana belum ada orang yang menyeberang jembatan. Simpul akar ini diberi label 1. Simpul 1 sekarang menjadi simpul hidup sekaligus simpul-E. Simpul 1 diperluas dengan meng-assign sisi (1,2) dengan (x_1,x_2) = (8,12), dengan arti yang menyeberang adalah orang dengan kecepatan 8 dan kecepatan 12 dengan t = 12 (sama dengan kecepatan orang yang lebih lambat). Sampai saat ini solusi adalah ((8, 12), x_3,(x_4, x_5), x_6, (x_7, x_8), x_9, (x_{10}, x_{11})). Solusi ini masih layak karena waktu yang diperlukan masih kurang dari M. Simpul 2 kemudian diperluas dengan meng-assign sisi (2,3) dengan x_3 = 12, artinya orang dengan kecepatan 12 kembali menyeberang untuk memberi lampu. sampai saat ini solusi adalah ((8, 12), 12,(x_4, x_5), x_6, (x_7, x_8), x_9, (x_{10}, x_{11})) dengan total waktu 24, masih memenuhi fungsi pembatas. Kemudian simpul 3 diperluas dengan mengg-*assign* sisi (3,4) dengan (x_4,x_5) = (6,12). Total waktu yang diperlukan adalah 36, tidak memenuhi fungsi pembatas, sehingga simpul 4 dibunuh, simpul 3 kemudian diperluas dengan membangkitkan anak-anak dari simpul 3. Jika tidak memenuhi fungsi pembatas, maka simpul tersebut dibunuh. Ketika tidak ada lagi simpul anak yang dapat dijadikan simpul-E, simpul 3 dibunuh kemudian simpul orangtua dari simpul 3 (simpul 2) dijadikan simpul-E berikutnya. Begitu seterusnya sampai solusi ditemukan atau tidak ada lagi simpul yang dapat digunakan untuk melakukan runut balik. Sebagian pohon yang dilahirkan dalam pencarian solusi permainan ini dapat dilihat pada gambar 3.

Pada gambar 2 dan gambar 3, simpul yang berwarna hijau adalah simpul yang dibangkitkan ketika dua orang menyeberang (dari kanan ke kiri), sedangkan simpul yang berwarna biru adalah simpul yang dibangkitkan ketika satu orang menyeberang (dari kiri ke kanan)



Gambar 2. Sebagian pohon ruang status persoalan *logic bridge*



Gambar 3. Sebagian pohon dinamis yang dibangkitkan dalam perncarian solusi permainan *logic bridge*

```
procedure logicbridge(input i,k: integer)
{mencari solusi penyeberangan jembatan dengan algoritma runut balik}

Deklarasi
 const maxtime = 30
 Type tabelkanan = array
 [1..5,1..5] of integer
 Type tabelkiri = array [1..5] of
 integer

 Kanan : tabelkanan
 {kanan adalah array orang yang
 berada di sisi kanan jembatan.
 Kanan[i,j] adalah waktu yang
 dibutuhkan ketika orang i dan j
 menyeberang}

 Kiri : tabel
 {kiri adalah array orang yang
 berada di sisi kiri
 jembatan.kiri[k] adalah waktu yang
 dibutuhkan orang k untuk
 menyeberang}

 T1(input i,j : integer)
 {fungsi pembangkit nilai
 kanan[i,j]}

 T2(input i : integer)
 {fungsi pebangkit nilai kiri[i]}

 B (input x : tabelsolusi)
 {fungsi pembatas untuk menentukan
 apakah x mengarah ke solusi}

 i,j,k : integer

Algoritma
j ← 1
for tiap kanan[i,j] yang belum
dicoba sedemikian hingga
kanan[i,j]← T1(i,j) dan B((x1,
x2),x3,…,(xi,xj)) = true do
  for tiap kiri[k] yang belum
  dicoba sedemikian hingga kiri[k]
  ← T2(k) dan  B((x1,
  x2),x3,…,xk,(xi,xj)) = true do
    if(((x1, x2),x3,…,xk,(xi,xj))
    adalah lintasan dari akar ke
    daun) then
       CetakSolusi(x)
    endif
    logicbridge(i,k+1)
  endfor
  j ← j+1
logicbridge(i+1,k)
endfor
```

Dalam program runut balik secara rekursif ini, tiap simpul dalam pohon ruang status berasosiasi dengan suatu pemanggilan rekursif.

Tiap simpul mempunyai kompleksitas n^2, sehingga jika pada pohon terdapat simpul sebanyak 2^n atau n!, maka untuk kasus terburuk, algoritma runut balik membutuhkan waktu dalam $O(n^2.2^n)$ atau $O(n^2.n!)$. [RM07]

Dengan algoritma runut balik ini diperoleh solusi ((1,3),3,(8,12),1,(1,6),1,(1,3)) dengan total waktu yang diperlukan untuk menyeberangi jembatan 3+3+12+1+6+1+3 = 29 detik.

3. KESIMPULAN DAN SARAN

Algoritma runut balik dapat digunakan dalam menyelesaikan permainan logic bridge. Namin dalam penyelesaiannya sebenarnya masih dapat dioptimasi (khususnya untuk membangkitkan daun) dengan menyimpan waktu menyeberang tiap orang dengan urutan menaik, sehingga jika daun dengan waktu menyeberang terkecil tidak menuju pada solusi, maka daun selanjutnya dari simpul tersebut tidak perlu dibangkitkan.

Karena tiap simpul dalam pohon ruang status memanggil fungsi rekursif ini, algoritma runut balik ini memiliki kompleksitas yang eksponensial

4. DAFTAR PUSTAKA

[RM07] Munir, Rinaldi. 2007. Diktat Kuliah IF2251 Strategi Algoritmik.