# Implementasi Algoritma A* Dalam Aplikasi Berbasis Web untuk Menemukan Rute Terpendek sebagai Navigasi Peta Digital Indoor

## *Implementation of A* Algorithm in Web-Based Applications for Finding the Shortest Route as Navigation of Digital Indoor Map*

**Afrizal Adam Maulana*[1], Wijanarto[2]**
Teknik Informatika UDINUS[1], Fakultas Ilmu Komputer UDINUS[2]
E-mail: ***[1]afrizalam1994@gmail.com, [2]wijanarto.udinus@gmail.com**

***Abstrak***

*Kemajuan teknologi dan fasilitas infrastruktur saat ini seringkali mengakibatkan seseorang sulit untuk mengingat semua jalan menuju ke tempat tertentu. Lebih dari 75% sebagian orang menghabiskan waktu dalam ruangan. Ketika mereka pergi ke tempat didalam bangunan dengan dimensi dan ukuran yang berbeda-beda, seperti, bangunan bertingkat, pusat perdagangan, pusat perbelanjaan, bandara, rumah sakit dan universitas, seringkali mereka bingung untuk menuju tempat yang diinginkan dalam waktu cepat dan tepat berdasarkan informasi yang terbatas. Pertanyaan yang akan dijawab dalam paper ini, bagiamana menganalisa prinsip dasar navigasi dan mendefinisikan instruksi dalam suatu domain model. Kemudian menentukan algoritma untuk menghasilkan rute terpendek. Terakhir mengimplementasikan algoritma dalam suatu aplikasi berbasis web dengan javascript. Solusinya adalah membuat model navigasi dengan unified modeling language peta digital, algoritma A-start dipilih sebagai solusi dalam menentukan rute untuk navigasi dalam peta serta membangun dan mengevaluasi aplikasi dengan javascript dan jasmine unit test. Hasil dari penelitian menunjukan bahwa model yang dipilih tepat untuk dapat diimplementasikan menjadi aplikasi berbasis web dan lolos evaluasi dengan 75 test case sebesar 100%, semua fungsionalitas aplikasi berjalan sesuai disain dan menghasilkan aplikasi tanpa anomaly.*

***Kata Kunci** — Navigasi, Indoor, Rute Terpendek, A-Star.*

***Abstract***

*Advances in technology and infrastructure facilities today often make it difficult for people to remember all the way to a particular place. More than 75% of people spend time indoors. When they go to places in buildings of different dimensions and sizes, such as, multilevel buildings, commercial centers, shopping centers, airports, hospitals and universities, they are often confused to get to the desired place in quick and precise time based on information limited. Questions to be answered in this paper, how to analyze the basic principles of navigation and define instruction in a domain model. Then determine the algorithm to generate the shortest route. Last implemented algorithm in a web based application with javascript. The solution is to create a navigation model with a unified modeling language digital map, A-start algorithm selected as a solution in determining the route for navigation in the map as well as build and evaluate applications with javascript and jasmine unit tests. The results of the research show that the chosen model is appropriate to be implemented into web-based applications and pass the evaluation with 75 test cases of 100%, all application functionality runs according to design and produces application without anomaly.*

***Keywords** — Navigation, Indoor, Shortest Route, A-Star.*


Dikirim: 14 Mei 2018; Direvisi: 28 Mei 2018; Diterima: 5 Juni 2018

## 1. PENDAHULUAN

Kemajuan teknologi dan fasilitas infrastruktur saat ini seringkali mengakibatkan seseorang sulit untuk mengingat semua jalan menuju ke tempat tertentu [1]. Manusia menghabiskan lebih dari 75% waktu mereka di dalam ruangan [2], oleh karena itu gerakan mereka jarang terbatas pada rute jalan dan terutama mengambil tempat di dalam bangunan dengan dimensi dan ukuran yang berbeda-beda. Dengan semakin kompleksnya lingkungan binaan tersebut, minat dalam memberikan panduan navigasi untuk ruang dalam ruangan semakin bertambah. Meskipun ada banyak cara untuk menuju ke lokasi tersebut seperti bertanya pada petugas ataupun orang lain, tetapi kita harus mengingat banyak arahan dan belum tentu kita sampai ke lokasi yang kita inginkan. Banyak data maupun peta dalam bentuk kertas yang telah diubah menjadi bentuk digital. Misalnya, Sebagai contoh, US Geological *Survey* (USGS) terus memindai dan merilis semua edisi lebih dari 200.000 halaman peta topografi bersejarah Amerika Serikat yang mencakup periode waktu 1884-2006. GIS Pusat di Academia Sinica, Taiwan, telah dipindai dan diarsipkan lebih dari 160.000 peta sejarah untuk didigitalisasi menjadi digital map [3].

Dari contoh di atas sebernarnya yang kita perlukan adalah suatu navigasi, yang dapat didefinisikan sebagai suatu arahan yang menunjukan pergerakan diri seseorang (satu tubuh) dalam suatu lingkungan [4]. Data peta untuk didalam bangunan indoor juga dibutuhkan dalam pencarian jalur, terutama orang yang pertama kali menginjak ke suatu area yang baru, Misalnya, ketika orang mengunjungi bangunan bertingkat dan kompleks seperti pusat perdagangan, pusat perbelanjaan, gedung pencakar langit, bandara, rumah sakit dan universitas, mereka memerlukan sistem navigasi untuk menemukan jalan mereka ke tempat tujuan [5]. Oleh karna itu peta digital menjadi suatu yang penting sebagai penunjuk arah dan pemilihan jalur menuju tempat tujuan yang diiginkan.

Paper ini akan mengangkat kasus Kampus Universitas Dian Nuswantoro mempunyai gedung-gedung terpisah dengan jumlah 8 gedung yaitu terdiri dari gedung A sampai dengan gedung H oleh sebab itu menyebabkan banyak orang bingung baik mahasiswa baru ataupun orang luar yang ada keperluan ke lokasi tertentu di kampus Universitas Dian Nuswantoro. Salah satu gedung yang sering menjadi sentral untuk aktivitas adalah gedung G dikarenakan gedung G merupakan tempat perkantoran utama Universitas Dian Nuswantoros dan juga sering dikunjungi oleh orang luar untuk keperluan kantor maupun yang lainnya. Setiap gedung terdapat fasilitas tangga manual, lift atau escalator, dan untuk pendatang untuk mencapai tujuan mereka tidak dibekali pengetahuan mengenai jalur mana yang terpendek atau tepat untuk mencapainya.

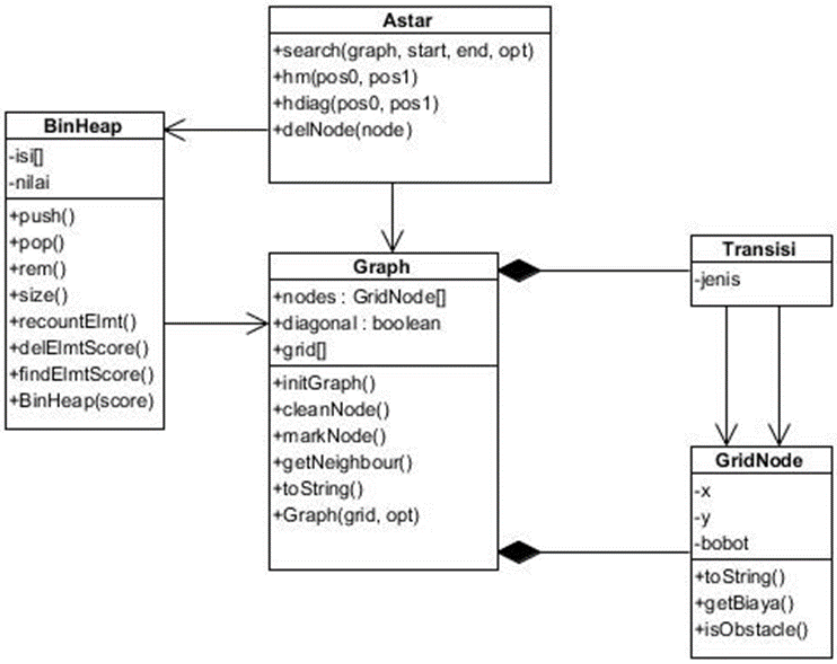
Dengan demikian maka dalam paper ini kami ingin menjawab semua persyaratan dengan menggabungkan proyek referensi yang dijelaskan sebelumnya, menangkap solusi terbaik untuk setiap kriteria. Sehingga pertanyaan utama yang akan dijawab dalam paper ini adalah, dapatkah pendekatan data visibilitas berguna untuk menghasilkan arahan rute pada level terendah dengan menggunakan suatu sistem yang dapat menemukan jalur terpendek dalam suatu ruangan.

## 2. METODE PENELITIAN

Dari pernyatan permasalahan sebelumnya, maka kita dapat menurunkan tujuan utama yang ingin dicapai paper ini yaitu, Pertama, menganalisa prinsip dasar navigasi untuk orang dan mendefinisikan instruksi dalam suatu domain model dengan UML (Unified Modeling Language) [6]. Kedua menentukan suatu algoritma pada level terendah sebagai instruksi yang menghasilkan suatu rute terpendek. Ketiga mengimplementasikan algoritma dalam suatu aplikasi berbasis web dengan javascript. Dalam rangka mengimplementasikan sistem yang akan dibuat, kami memiilih Universitas Dian Nuswantoro sebagai obyek riset sekaligus sebagai test prototype aplikasi, dengan demikian data model geometri bangunan yang dipergunakan dalam kasus ini adalah gedung G dari lantai 1 hingga 3 di Universitas Dian Nuswantoro.

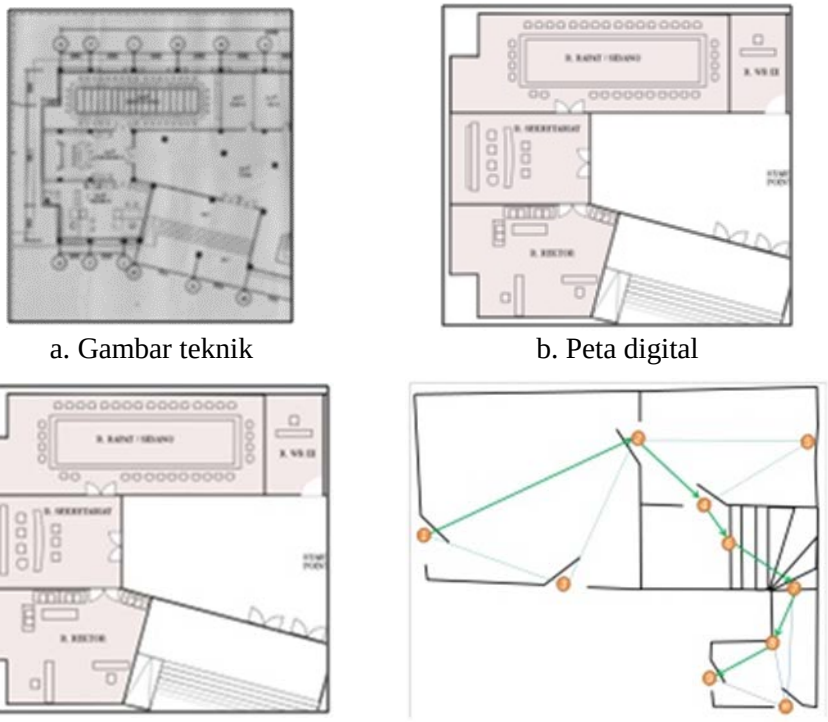
*2.1. Domain Model Navigasi*

Model domain mewakili konsep utama domain, mengidentifikasi hubungan antara semua entitas dan mencantumkan atribut mereka, memberikan pandangan struktural tentang domain, menjelaskan dan membatasi cakupan domain masalah. Model domain dapat digunakan untuk memverifikasi dan memvalidasi pemahaman tentang masalah domain di antara berbagai pemangku kepentingan (stakeholder). Ini menggambarkan kosa kata dan sangat membantu sebagai alat komunikasi. Paper ini akan mengadopsi model door to door dengan representasi graph [7,8,9] yang dimodelkan dengan UML.




Gambar 1. Model UML untuk navigasi lingkungan

Model yang disajikan dalam Gambar 1 di atas dapat di jelaskan, bahwa lingkungan peta didasarkan atas GridNode yang terdiri dari pintu, sudut ruangan, tangga (sebagai pintu virtual), mebel yang mewakili suatu node dalam graph. Transisi adalah garis yang menghubungkan antar node dalam graph. Semua node akan disimpan dalam struktur data heap yang akhirnya dipakai sebagai input untuk menentukan rute terpendek oleh obyek astar yang menghasilkan path dari posisi awal sampai tujuan yang dimaksud. Berikut proses pemodelan yang akan dibuat dari data mentah menjadi data logic (struktur data) yang dapat dikomputasikan seperti disajikan pada Gambar 2.a, yaitu data mentah dalam bentuk gambar teknik, yang akan diekstrak menjadi peta digital (Gambar 2.b) yang terdiri dari *base*, rooms, start point, end point, lines, door dan path. Setelah dimodelkan dan diberi tanda akan disimpan dalam format low level dalam bentuk graph (Gambar 2.c) yang terdiri dari node (lingkaran merah), yang mewakili pintu, pintu virtual (tangga/lift/escalator), sudut ruang, mebel dan garis transisi (edge, garis hijau) seperti disajikan berikut pada Gambar 2, kemudian hasil akhirnya berupa bentuk logis yang dapat diolah oleh computer (struktur data).



a. Gambar teknik b. Peta digital

c. Graph dalam peta d. Path pencarian

Gambar 2. Representasi Data Model, Peta, Graph dan Path

Tampak garis solid hijau (Gambar 2.d) dengan tanda panah adalah path yang dihasilkan oleh algoritma yang melalui titik-titik transisi (oranye) yang dimodelkan secara berurutan dari 1-2-4-5-7-8-9, 1 adalah titik awal, dan 9 adalah titik akhir.

*2.2. Algoritma A-star*

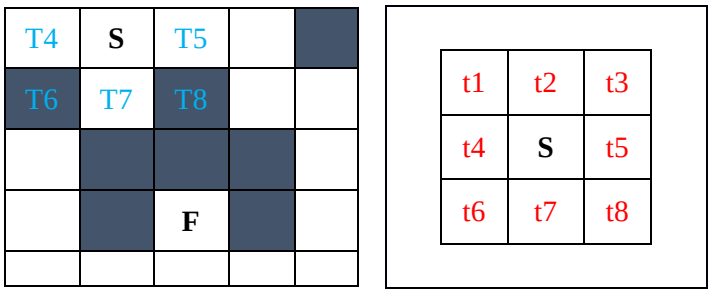
Algoritma jalur optimal yang umum digunakan biasanya dibagi menjadi dua kategori berikut: algoritma optimal dan algoritma heuristik. Algoritma optimal dapat menjamin solusi optimal dari masalah jalur optimal, tetapi kompleksitas waktunya meningkat dengan ukuran jaringan secara eksponensial dan diperlukan untuk mendesain ulang strategi pencarian ketika diterapkan pada sistem navigasi waktu nyata [10] algoritma heuristik tidak dapat menjamin solusi optimal, tetapi sering dapat menemukan solusi yang layak sub-optimal dalam waktu yang terbatas [11]. Bahkan, ada banyak algoritma pencarian heuristik, seperti metode pencarian preferensi lokal [12], algoritma Best first search [13] dan algoritma A* [14], berdasarkan studi pustaka di atas paper ini memilih algoritma A* karena solusi optimal tercapai walaupun waktu yang diperlukan kurang efisien.

Penerapan navigasi (wayfinding) digital dengan algoritma A-star [15,16] dan merupakan perbaikan dari Dijkstra [17], yang ditentukan oleh persamaan f(n)=g(n)+h(n), dimana f(n) adalah jumlahan dari g(n) Geographical Cost, g(n) = jarak dari posisi sebelum ke posisi sekaran dan h(n) Heuristic Cost, adalah jarak dari posisi sekarang ke posisi akhir (finish) dengan manhattan distance. Misalkan, jika diketahui x= (a, b) dan y= (c, d), maka manhattan distance dihitung dengan persamaan 1.

$$|a - c| + |b - d| \qquad (1)$$

dimana: x = langkah yang mendatar (kanan/kiri)
y = langkah yang menurun (atas/bawah)

Berikut contoh perhitungan algoritma A* dengan matrik seperti Gambar 3.a dan 3.b sebagai berikut, misal terdapat map dengan titik awal S dan akhir F seperti Gambar 3.a yang terdiri dari kotak warna putih dan biru tua, yaitu tembok atau obstacle atau halangan yang tidak dapat dilewati, langkah awal adalah menentukan kernel 8 tetangganya (Gambar 3.b).




a. Map awal b. 8 tetangga

Gambar 3. Map dan tetangganya

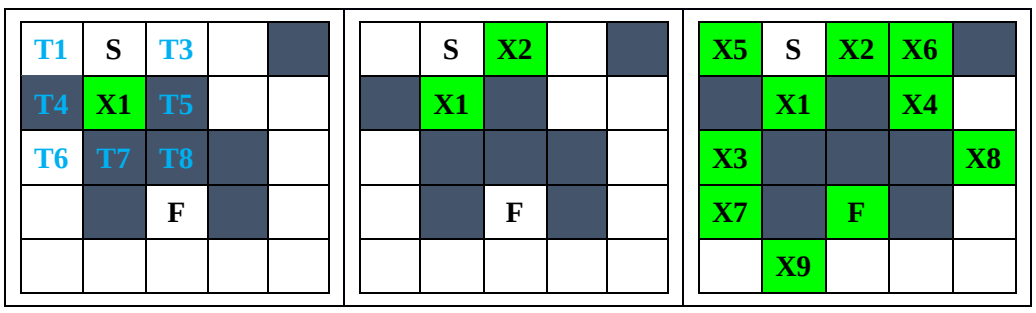
Dari map di atas untuk titip S, hanya terdapat *t4*, *t5*, *t6*, *t7*, dan *t8* yang merupakan titik tetangga dengan pusat S. Karena titik S berada pada pinggir map sehingga tidak memiliki tetangga di atasnya (*t1*, *t2*, *t3*), sehingga kita tidak bisa melewati jalur tersebut. Dengan demikian, tetangga yang akan kita dipertimbangkan untuk dihitung atau operasikan adalah *t4*, *t5* dan *t7* saja, karena *t6* dan *t8* berupa tembok atau halangan. Selanjutnya kita akan mengitung g(n), h(n) dan f(n) dari masing-masing tetangganya (*t4*, *t5*, *t7*) dan menentukan parentnya.

**Menghitung f(n) pada t4** : g(n) dihitung dari titik S ke titik *t4*, *x* kekanan 1 satuan, dan *y* tidak bergerak berarti 0 satuan, g(n) = $|a-c|+|b-d|$= 1+0 = 1. Selanjutnya nilai h(n) dihitung dari titik *t4* ke titik F, yaitu x= kekanan 2 satuan dan y kebawah 3 satuan, h(n) =$|a-c|+|b-d|$ = 2+3 = 6, sehingga f(n) = g(n) + h(n) = 1 + 6 = 7, jadi f(n) dari *t4* = 7 dengan parent S.

**Menghitung f(n) pada t5** : g(n) dihitung dari titik S ke titik *t5*, *x* kekiri 1 satuan, dan *y* tidak bergerak berarti 0 satuan, g(n) = $|a-c|+|b-d|$= 1+0 = 1, Selanjutnya nilai h(n) dihitung dari titik *t5* ke titik F, yaitu x tidak bergerak = 0 satuan dan y kebawah 3 satuan, h(n) =$|a-c|+|b-d|$ = 0+3 = 3, sehingga f(n) = g(n) + h(n) = 1 + 3 = 4, jadi f(n) dari *t5* = 4 dengan parent S.

**Menghitung f(n) pada t7** : g(n) dihitung dari titik S ke titik *t7*, *x* kekanan 1 satuan, dan *y* tidak bergerak berarti 0 satuan, g(n) = $|a-c|+|b-d|$= 0+1 = 1. Selanjutnya nilai h(n) dihitung dari titik *t7* ke titik F, yaitu x= kekanan 1 satuan dan y kebawah 2 satuan, h(n) =$|a-c|+|b-d|$ = 1+2 = 3, lalu f(n) = g(n) + h(n) = 1 + 3 = 4, jadi f(n) dari *t7* = 4 dengan parent S.

Setelah mendapatkan f(n) dan parent dari *t4*, *t5* dan *t7*, kita cari hasil f(n) yang paling kecil dari *t4*, *t5*, *t7*. Dan kita akan menuju ke titik tersebut (misal *t7* diberi tanda dan warna X1). Dari hasil langkah 2, diketahui: f(n) *t4* = 7, parent S; f(n) *t5* = 3, parent S; f(n) *t7* = 3, parent S lalu kita ambil f(n) yang paling kecil yaitu *t7*, maka kita menuju ke titik *t7* dan tandai dengan X1. Lalu hitung f(n) pada X1= g(n) + h(n)= 1 + 3= 4. Hal tersebut dilakukan terus menerus sehingga hasil akhir diperoleh pada gambar 4.a, b dan c di bawah ini.




iterasi ke-1 iterasi ke-2 iterasi ke-9

Gambar 4. Iterasi proses perhitungan A*

Dari hasil di atas, maka jarak terpendek dari S ke F adalah S - X1 – X3 – X7 – X9 – F. Untuk pencarian jalur pada path didalam peta digital memerlukan penetapan aturan dasar yang terdiri dari *base* sebagai dasaran, pintu (door), titik (node), jalur (path), dan penghubung antar *base* (portal). *Base* merupakan sketch dasar gambar yang terdiri dari ruangan-ruangan yang berada di map. Per *base* berbeda-beda dari node maupun pathnya. *Node* adalah titik puncak, dan *path* berperan sebagai *arcs* yang berfungsi sebagai garis penghubungnya antar *node*.

### *2.3. Pembangunan Sistem dan Notasi Algoritmik*

Pembuatan aplikasi ini berbasis web yang menggunakan tekniik V-model Software Development Live Cyrcle yang dikodekan dengan *javascript* dan *map SVG*. Berikut metode terpilih akan mengerjakan urutan langkah yaitu, proses scanning data gambar peta, digitalisasi gambar peta, pengelompokan *Rooms, Paths, Doors, Portal*, pengkodean, terakhir validasi atau tesing. Berikut notasi algoritmik untuk fungsi utama penentu rute terpendek berdasarkan [18] dengan penyesuaian, testing dilakukan dengan *jasmine unit testing* yang dirancang untuk melihat *behavior driven development*.

**Notasi A-star**

```
search(graph, awal, akhir, options)
  cleanGraph();    options ← options Or {};
  heuristic← options.heuristic Or astar.heuristics.manhattan;
  closest ← options.closest Or | false;
  openHeap ← getHeap();
  closestNode ← awal;
  awal.h ← heuristic(awal, akhir);
  graph.markDirty(awal);
  openHeap.push(awal);
  while (openHeap.size() > 0) do
    currentNode ← openHeap.pop();
  if (currentNode === akhir) then
   → pathTo(currentNode);
  currentNode.closed ← true;
  neighbors ← graph.neighbors(currentNode);
  for (i← 0, il←neighbors.length; i < il; ++i) do
    neighbor← neighbors[i];
  if (neighbor.closed || neighbor.isWall()) then.
    continue;
  gScore← currentNode.g+ neighbor.getCost(currentNode);
    beenVisited ← neighbor.visited;
if (!beenVisited || gScore < neighbor.g) then
  neighbor.visited← true; neighbor.parent ← currentNode;
  neighbor.h ← neighbor.h OR heuristic(neighbor, akhir);
  neighbor.g ← gScore;
  neighbor.f ← neighbor.g + neighbor.h;
  graph.markDirty(neighbor);
  if (closest) then
    if (neighbor.h < closestNode.h || (neighbor.h =
      closestNode.h && neighbor.g < closestNode.g)) then
      closestNode← neighbor;
    if (!beenVisited) then
      openHeap.push(neighbor);
    else
      openHeap.rescoreElement(neighbor);
    if (!beenVisited) then
      openHeap.push(neighbor);
    else
      openHeap.rescoreElement(neighbor);
    if (closest) {
     → pathTo(closestNode);
  → return [];
```

**Notasi Heuristik**

```
manhattan(c)
  d1←abs(pos1.x - pos0.x);d2←abs(pos1.y - pos0.y);
  → d1 + d2;
```

**Notasi diagonal movement**

```
diagonal(pos0, pos1)
   D ← 1;  D2 ←sqrt(2);d1 ←abs(pos1.x - pos0.x);
   d2 ←abs(pos1.y - pos0.y);
   → (D * (d1 + d2)) + ((D2 - (2 * D)) *min(d1, d2));
```

**Notasi Algoritmik penentuak tetangga**

```
neighbors(node)
  ret← [];  x ← node.x;  var y ← node.y;
  grid ← this.grid;
  if (grid[x - 1] && grid[x - 1][y]) then
    ret.push(grid[x - 1][y]);
  if (grid[x + 1] && grid[x + 1][y]) then
    ret.push(grid[x + 1][y]);
  if (grid[x] && grid[x][y - 1]) then
    ret.push(grid[x][y - 1]);
  if (grid[x] && grid[x][y + 1]) then
    ret.push(grid[x][y + 1]);
if (this.diagonal) then
    if (grid[x - 1] && grid[x - 1][y - 1]) then
      ret.push(grid[x - 1][y - 1]);
    if (grid[x + 1] && grid[x + 1][y - 1]) then
      ret.push(grid[x + 1][y - 1]);
    if (grid[x - 1] && grid[x - 1][y + 1]) then
      ret.push(grid[x - 1][y + 1]);
    if (grid[x + 1] && grid[x + 1][y + 1]) then
      ret.push(grid[x + 1][y + 1]);
  → ret;
```

Aplikasi akan ditest dengan 2 teknik testing, pertama, blackbox testing terhadap fungsionalitas aplikasi. Kedua Unit testing (Behavior Driven Development) dengan tool jasmine dan scenario seperti dipaparkan pada potongan kode berikut.

```
describe('Test Behavior Driven development Algoritma AStar', function () {
 /*10X10*/
 var graphIn1=[[0,1,1,1,1,1,1,1,1,1],
               [1,1,0,0,1,0,1,1,1,0],
               [1,0,0,1,1,1,1,1,1,1],
               [1,1,1,1,1,0,1,0,0,1],
               [1,1,1,1,1,1,1,1,1,1],
               [1,1,1,1,1,1,1,1,1,0],
               [1,1,1,1,1,1,1,1,0,1],
               [1,1,1,1,1,1,1,0,0,1],
               [0,1,1,1,1,1,1,1,0,1],
               [1,0,1,1,1,1,1,1,1,1]];
 describe('Test Graph 1 (10X10) - Path finding', function () {
  var aw1="[0,1]";var ak1="[9,9]";
  it("variasi     1:     awal"+aw1+":akhir"+ak1+"...hasil     sesuai     yang
diharapkan.",function(){
       var posAwal1=[0,1];
       var posAkhir1=[9,9];
       var result1 = runSearch(graphIn1, posAwal1, posAkhir1);
    expect("(0,2),(0,3),(0,4),(1,4),(2,4),(2,5),(2,6),(3,6),
            (4,6),(5,6),(6,6),(7,6),(8,6),(9,6),(9,7),(9,8),
            (9,9)").toEqual(result1.text);
 });
…
});
```

Pada Tabel 1, disajikan rencana data input **graph** yang direpresentasikan dalam suatu **matrik** 10X10, 15X15 dan 20X20, dengan masing-masing terdapat 5 variasi pada obstacle dan 5 variasi pada node awal dan akhir.

Tabel 1. Test case input untuk algoritma A Start

| **Dimensi Graph** | **Variasi obstacle/dinding** | **Variasi node awal dan akhir** |
|---|---|---|
| 10X10 | 5 variasi | 5 variasi |
| 15X15 | 5 variasi | 5 variasi |
| 20X20 | 5 variasi | 5 variasi |

Sementara Gambar 5.a. entri matrik bernilai 1 menyatakan node terbuka atau jalan dan 0 menyatakan node tertutup atau dinding atau obstacle dengan koordinat node awal (Gambar 5.b) sebagai titik awal pencarian dan koordinat node akhir sebagai titik akhir pencarian.

```
[0,1,1,1,1,1,1,1,1,1]
[1,1,0,0,1,0,1,1,1,0]
[1,0,0,1,1,1,1,1,1,1]
[1,1,1,1,1,0,1,0,0,1]
[1,1,1,1,1,1,1,1,1,1]
[1,1,1,1,1,1,1,1,1,0]
[1,1,1,1,1,1,1,1,0,1]
[1,1,1,1,1,1,1,0,0,1]
[0,1,1,1,1,1,1,1,0,1]
[1,0,1,1,1,1,1,1,1,1]
```

```
[[1,1,1,1,1,1,1,1,0,1,1,1,1,1,1],
 [0,1,1,1,1,1,1,1,1,1,1,1,0,0,1],
 [1,1,1,1,1,1,1,1,1,1,1,1,1,0,1],
 [1,1,0,1,1,1,1,1,0,1,1,1,0,1,1],
 [1,1,1,0,1,0,1,1,1,1,1,1,1,1,0],
 [0,1,0,1,1,1,0,1,1,1,1,1,1,1,1],
 [1,1,1,1,1,1,1,0,1,1,1,0,1,1,1],
 [1,1,0,1,0,1,1,1,1,1,1,0,1,1,1],
 [1,1,0,1,1,1,1,1,1,0,1,1,1,1,1],
 [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
 [1,1,1,1,1,1,0,1,1,1,0,1,0,1,1],
 [1,1,0,1,1,1,1,1,1,1,1,1,1,1,1],
 [1,1,0,1,1,0,1,1,1,1,1,1,1,1,1],
 [0,1,1,1,1,1,1,1,1,1,1,1,1,1,1]]
```

```
[[1,1,1,1,1,1,0,1,1,1,1,1,1,0,1,1,1,1,1,1],
 [1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1],
 [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,0],
 [1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,0],
 [0,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1],
 [1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1],
 [1,1,0,1,1,0,1,1,1,1,1,1,1,1,1,0,1,1,1,1],
 [1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
 [1,1,1,1,0,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0],
 [1,1,1,1,0,1,1,1,1,1,1,1,1,1,0,1,1,0,1,1],
 [1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1],
 [1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1],
 [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
 [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
 [1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1],
 [1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,0,1,1],
 [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
 [1,1,1,1,1,1,1,1,0,1,0,1,0,1,1,1,1,1,1,1],
 [1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,0,1,1],
 [0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]]
```

a. Input Graph

```
var posAwal1= [0,1];
var posAkhir1= [9,9];
var posAwal2= [0,4];
var posAkhir2= [8,2];
var posAwal3= [3,0];
var posAkhir3= [0,9];
var posAwal4= [9,9];
var posAkhir4= [0,9];
var posAwal5= [9,5];
var posAkhir5=[0,5];
```

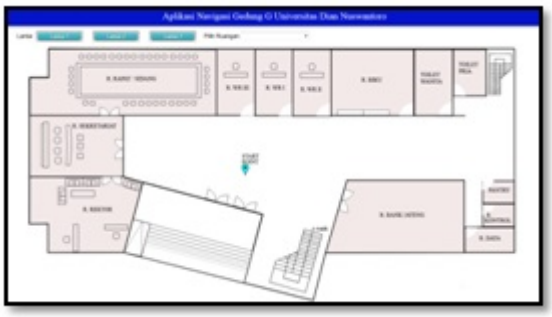
b. Node Awal Dan Akhir

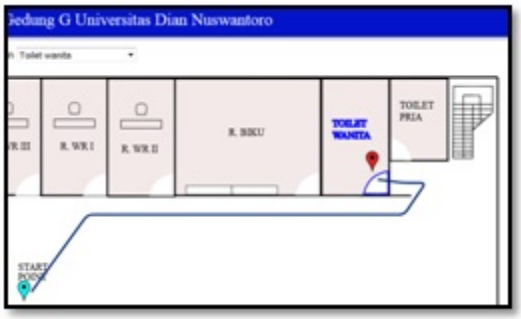
Gambar 5. Skenario Test Input dan Output

Input data akan divariasi baik dari ***dimensi*** matrik, ***sebaran*** dinding (obstacle), ***node awal*** dan ***node akhir***, kita akan melakukan setidaknya 75 test untuk 3 jenis ukuran matrik (10X10, 15X15 dan 20X20), 5 kombinasi variasi obstacle dan 5 variasi node awal dan akhir.

## 3. HASIL DAN PEMBAHASAN

Paper ini berhasil mengimplementasikan algoritma A* menjadi suatu aplikasi navigasi dalam gedung (indoor wayfinding) untuk mendapatkan rute terpendek dari posisi awal menuju akhir sesuai yang diharapkan seperti terlihat pada gambar 6 di bawah ini.






1. Aplikasi Navigasi Gedung G 2. Path *User*-Toilet Wanita

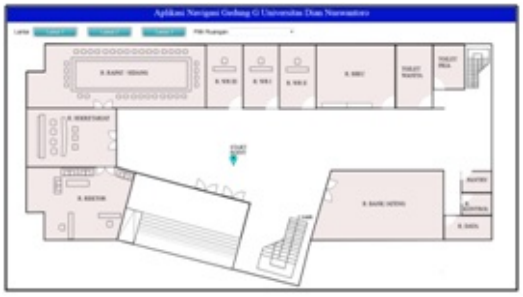
Gambar 6. Aplikasi Utama dan contoh path yang terbentuk

Hasil dari blackbox test yang telah dilakukan disajikan pada Tabel 2, dimana semua fungsional menu memberikan respon sesuai disain dan tidak terdapat anomaly didalamnya.

Tabel 2. Black Box Testing Aplikasi Navigasi

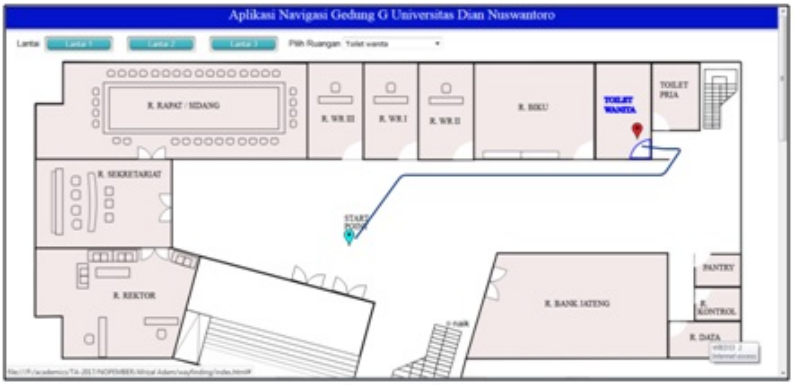
| No | Judul | Ekspektasi | Hasil |
|---|---|---|---|
| 1 | Button 1st Floor | Terdapat tombol 1st Floor untuk mengakses map lantai 1 | Berhasil |
| 2 | Button 2nd Floor | Terdapat tombol 2nd Floor untuk mengakses map lantai 2 | Berhasil |
| 3 | Button 3rd Floor | Terdapat tombol 3rd Floor untuk mengakses map lantai 3 | Berhasil |
| 4 | Combo Box pilihan end point | Terdapat tombol combobox yang menampilkan pilihan end point di setiap lantai | Berhasil |
| 5 | Maps floor1 | Terdapat map lantai 1 dan siap ditampilkan | Berhasil |
| 6 | Maps floor2 | Terdapat map lantai 2 dan siap ditampilkan | Berhasil |
| 7 | Maps floor3 | Terdapat map lantai 3 dan siap ditampilkan | Berhasil |
| 8 | Respon Mouse Event | Aplikasi dapat merespon mouse event untuk menampilkan path dari end point yang dipilih dengan mouse. | Berhasil |

Berikut contoh screen shoot dari uji Tabel 2 di atas (tidak semua ditampilkan), Gambar 7.a respon tombol lantai dan map yang tampil, Gambar 7.b respon combobox terpilih untuk menentukan end point dan Gambar 7.c respon mouse event saat end point di pilih dengan mouse dan menghasilkan path.



a. Button Lantai 1 Terpilih

b. Button Combox terpilih




c. Mouse click eveevent

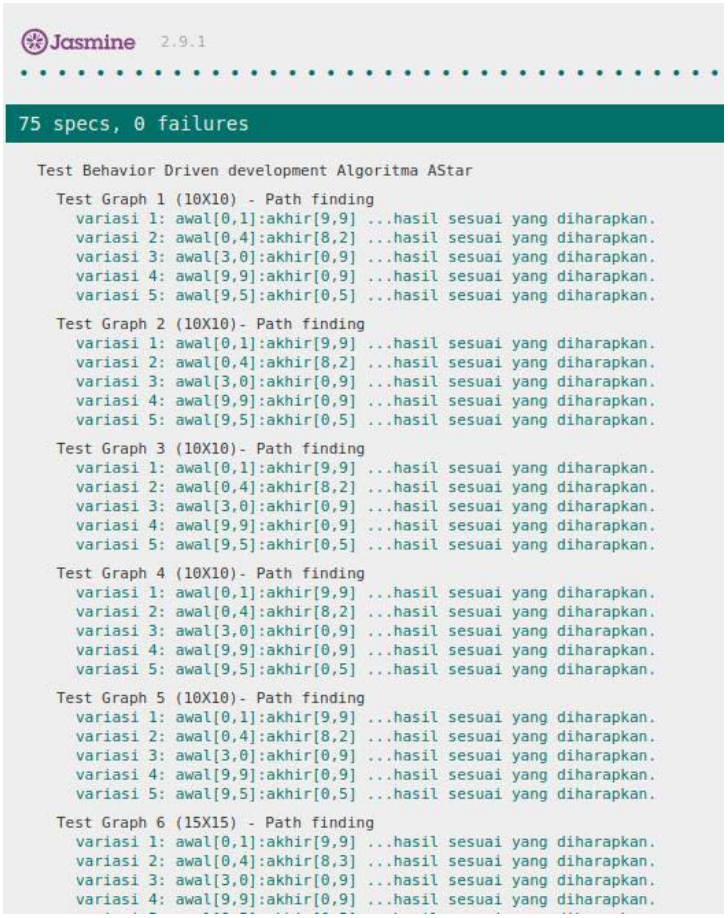
Gambar 7. Screen shoot hasil Blacbox Testing

Sementara itu hasil evaluasi unit test dengan jasmine terhadap 75 test case yang dikerjakan dengan variasi input dimensi graph, sebaran obstacle, sebaran titik awal dan akhir, semuanya dapat menghasilkan **path** sesuai scenario yang diharapkan, berikut Tabel 3 menyajikan hasil testing dari **input graph**, **input** node **awal** dan **akhir**, serta **output path** yang dihasilkan. Hasil evaluasi ini tentunya merupakan perbaikan atas apa yang belum dilakukan dalam [17].

Tabel 3. Skenario Unit testing Algoritma A Star

| Graph | Variasi Node Awal dan Akhir | Path |
|---|---|---|
| 10X10 | Variasi 1 | (0,2),(0,3),(0,4),(1,4),(2,4),(2,5),(2,6),(3,6),(4,6),(5,6),(6,6),(7,6),(8,6),(9,6),(9,7),(9,8),(9,9) |
| | Variasi 2 | (1,4),(2,4),(2,3),(3,3),(4,3),(4,2),(5,2),(6,2),(7,2),(8,2) |
| | Variasi 3 | (2,0),(1,0),(1,1),(0,1),(0,2),(0,3),(0,4),(0,5),(0,6),(0,7),(0,8),(0,9) |
| | Variasi 4 | (9,8),(9,7),(8,7),(8,6),(7,6),(6,6),(5,6),(4,6),(3,6),(2,6),(2,7),(2,8),(1,8),(0,8),(0,9) |
| | Variasi 5 | (8,5),(7,5),(6,5),(5,5),(4,5),(4,6),(3,6),(2,6),(1,6),(0,6),(0,5) |
| 15X15 | Variasi 1 | (0,2),(0,3),(0,4),(1,4),(2,4),(2,5),(2,6),(3,6),(4,6),(5,6),(6,6),(7,6),(8,6),(9,6),(9,7),(9,8),(9,9) |
| | Variasi 2 | (1,4),(2,4),(3,4),(4,4),(5,4),(6,4),(6,3),(7,3),(8,3) |
| | Variasi 3 | (2,0),(2,1),(2,2),(2,3),(2,4),(2,5),(2,6),(2,7),(2,8),(2,9),(1,9),(0,9) |
| | Variasi 4 | (8,9),(7,9),(6,9),(5,9),(4,9),(3,9),(2,9),(1,9),(0,9) |
| | Variasi 5 | (8,5),(7,5),(6,5),(5,5),(5,4),(4,4),(3,4),(2,4),(2,5),(1,5),(0,5) |
| 20x20 | Variasi 1 | (0,2),(1,2),(1,3),(2,3),(3,3),(4,3),(5,3),(6,3),(6,4),(7,4),(7,5),(8,5),(8,6),(9,6),(9,7),(9,8),(9,9) |
| | Variasi 2 | (1,4),(1,3),(2,3),(3,3),(4,3),(5,3),(6,3),(7,3),(8,3),(8,2) |
| | Variasi 3 | (2,0),(2,1),(2,2),(2,3),(1,3),(1,4),(1,5),(1,6),(1,7),(1,8),(0,8),(0,9) |
| | Variasi 4 | (8,9),(7,9),(6,9),(5,9),(4,9),(3,9),(2,9),(1,9),(0,9) |
| | Variasi 5 | (8,5),(7,5),(7,6),(6,6),(5,6),(4,6),(4,5),(3,5),(2,5),(1,5),(0,5) |

Pada Gambar 8 disajikan screen shoot hasil evaluasi dan ringkasan test yang telah dilakukan dengan jasmine, berdasarkan inputan seperti pada Gambar 5 di atas.




Gambar 8. Evaluasi dengan Jasmine BDD unit testing untuk algoritma A-Star

## 4. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan dan disajikan pada bagian sebelumya, maka paper ini dapat menimpulkan hasil penelitin sementara seperti berikut di bawah ini:

1. Aplikasi berbasis web untuk menemukan rute terpendek sebagai navigasi peta digital indoor berhasil dibangun dengan menerapkan algoritma A-star untuk menentukan jalur terpendek dari awal hingga akhir.
2. Testing dan evaluasi yang dilakukan dengan tool jasmine terbukti bahwa algoritma yang diimplementasikan berhasil menghasilkan rute terpendek dari sebaran dan variasi input yang diberikan sebesar 75 test case, terbukti dapat berhasil 100% lolos test dan evaluasi. Hasil evaluasi ini sekaligus merupakan perbaikan atas [17] yang belum melakukan evaluasi dan test unit secara empiris.

## 5. SARAN

Keberhasilan penerapan algoritma A-star ke dalam aplikasi berbasis web untuk menghasilkan rute terpendek untuk navigasi dalam gedung, masih perlu penyempurnaan, untuk itu dapat disarankan sebagai berikut:

1. Aplikasi disarankan untuk dapat dikembangkan berbasis mobile (Android, IOS dan lainnya) dengan memanfaatkan *access point* sebagai *virtual door* untuk lingkungan *indoor* dan *Base Transceiver Station (BTS)* untuk *outdoor*.
2. Perlu kajian mendalam mengenai algoritma yang tepat untuk diimplementasikan baik untuk *indoor* dan *outdoor navigation*, serta pemanfaatan infrastruktur Information Technology (IT) di lingkungan yang dipilih.

## DAFTAR PUSTAKA

[1] Vaidya, A., Kumbhar, C., Yeole. K., and Rasal, S., 2014, Indoor Guidance for Public Buildings using Android Smartphones, *International Journal of Engineering Science & Advanced Technology*, Vol. 4, No. 1, Hal 68 – 71

[2] Klepeis, N. E., Nelson, W. C., Ott, W. R., Robinson, J. P., Tsang, A. M., Switzer, P., Behar, J. V., Hern, S.C., Engelmann, W. H., 2001, The National Human Activity Pattern *Survey* (NHAPS): a resource for assessing exposure to environmental pollutants. *Journal of exposure analysis and environmental epidemiology*, Vol. 1, No. 3, Hal 231 - 252.

[3] Chiang, Y. Y., Knoblock, C. A., Leyk, S., 2014, A *Survey* of Digital Map Processing Techniques, *ACM Computing Surveys*, Vol. 47, No. 1.

[4] Montello, D. R., 2005, *Navigation. In Cambridge University, editor, The Cambridge Handbook of Visuospatial thinking*, Cambridge University Press.

[5] Ortakci, Y., Karas, R. I., Rahman, A. A., 2014, 3D Indoor Navigation Prototype for Smartphones, *3DGeoInfo 2014, Conference*, Dubai, November 12-13.

[6] Balaji, S., Murugaiyan, D. S., 2012, WATEERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC, *International Journal of Information Technology and Business Management*, Vol. 2, No. 1, Hal 26 – 30.

[7] Russo, D., 2012, Route Directions using Visible Landmarks for an Indoor Navigation System based on Android device: \IndoorNav, *Thesis*, Università degli Studi dell'Aquila – Facoltà, Ingegneria.

[8] Liu, L., Zlatanova, S., 2011, A door-to-door path-fnding approach for indoor navigation. *In Proceedings of GeoInformation for Disaster Management Conference 2011*, Turkey, 3-8 Mei 2011.

[9] Ali, H. M., Noori, Z. T., 2016, Indoor Way Finder Navigation System Using Smartphone, *IJCSMC*, Vol. 5, No. 1, Hal 202 – 218.

[10] Qiuping, W., Rong, S., Qi, Z., 2013 Optimal path selection of slow traffic based on GIS network analysis, *Journal of Xi'an University of Architecture & Technology*, Vol. 45, No. 5, Hal 668-674.

[11] Lixiao, L., 2013 A Catastrophe Operator Based on Logistics Lines Optimization Algorithm, *Bulletin of Science and Technology*, Vol. 29, No. 8, Hal 214 - 216.

[12] Xia, L., Hao, P., 2015 Load Balancing in Cloud Computing Using Stochastic Hill Climbing-A Approach, *Modem Computer*, Vol. 2015, No. 19, Hal 3 - 7.

[13] Jianxu, Z., Yan, J., 2015 Determining Effective Paths Set Based on Reverse Depth-First Search Algorithm, *Journal of Chongqing Jiaotong Un iversity*, Vol. 34, No. 3, Hal 93-98.

[14] Renhao, X., LiuYu, 2015, Improvement and parallelization of A* algorithm, *Journal of Computer Applications*, Vol. 35, No. 7, Hal 1843 - 1848.

[15] Goyal, A., Mogha, P., Luthra, R., Sangwan, M. N., 2014, Path Finding: A* or DIJKSTRA'S, *International Journal in IT and Engineering*, Vol. 2, No. 1, Hal 1 – 15.

[16] Rahmanto, A. F., Wijanarto, W., 2017, Penemu Jalur Optimal Untuk Rute Jalan Dengan New Bidirectional A* Di Semarang, *Techno.Com*, Vol. 16, No. 2, Hal 182 – 194.

[17] Maulina, S., 2016, Penerapan Algoritma Dijkstra untuk Menentukan Jalur Terpendek pada Aplikasi Digital Wayfinding di Universitas Dian Nuswantoro Semarang Berbasis Web, *http://eprints.dinus.ac.id/18750/*, diaskses 05-12-2017

[18] Brian, 2012, A* Search Algorithm in JavaScript, *https://briangrinstead.com/blog/astar-search-algorithm-in-javascript-updated/* diakses tanggal 15 November 2017.