Pemilihan Lokasi Pertemuan Optimal Menggunakan Algoritma *Divide and Conquer*

Felix Septianus Darmawan 13516041
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10, Bandung, Indonesia
13516041@std.stei.itb.ac.id

***Abstraksi*—Lokasi pertemuan optimal adalah lokasi yang meminimalkan jumlah jarak dari lokasi pertemuan optimal ke setiap lokasi lainnya. Minimasi jarak dapat menghemat waktu dan biaya untuk mencapai lokasi. Makalah ini membahas algoritma yang efisien untuk mendapatkan lokasi pertemuan yang optimal. Jarak yang digunakan untuk perhitungan adalah jarak Manhattan. Lokasi pertemuan optimal dapat didapatkan dari median geometri dari titik-titik lokasi.**

Kata kunci—lokasi; minimal; jarak

I. PENDAHULUAN

Dalam kehidupan sosial manusia, kita sering memiliki kegiatan yang membutuhkan koordinasi dengan banyak pihak, seperti pengerjaan tugas kelompok atau rekreasi bersama. Dalam kegiatan tersebut, dibutuhkan pemilihan lokasi berkumpul yang adil untuk seluruh pihak karena perlu dipikirkan biaya yang diperlukan untuk mencapai lokasi berkumpul. Terkadang, orang kesulitan menentukan lokasi tersebut karena ada beberapa pihak yang merasa lokasi yang dipilih kurang adil untuk pihak tertentu. Namun setiap pihak dapat bias dalam menentukan lokasi tersebut, sehingga sulit mencapai kesepakatan.

Pemilihan lokasi tersebut dapat dihitung dari jarak setiap rumah pihak yang terlibat. Lokasi yang dipilih adalah lokasi yang memiliki jumlah jarak minimal dari lokasi yang dipilih ke semua lokasi lain. Lokasi tersebut dianggap adil jika tidak ada lokasi lain yang mendekatkan jarak dari sebagian pihak lebih besar daripada menjauhkan jarak dari sebagian pihak yang lain. Dengan memilih lokasi yang optimal sebagai lokasi pertemuan, pihak yang terlibat dapat menghemat biaya transportasi dan waktu untuk mencapai lokasi pertemuan.

Dewasa ini, kehidupan perkotaan semakin meluas ke semua daerah. Jalan raya selalu ditemukan di perkotaan. Jalan raya memiliki persimpangan sebagai titik pertemuan dengan jalan raya lainnya. Persimpangan di jalan raya umumnya membentuk sudut sembilan puluh derajat. Dengan asumsi tersebut, pengukuran jarak antara dua lokasi di perkotaan dapat didekati dengan jarak manhattan.

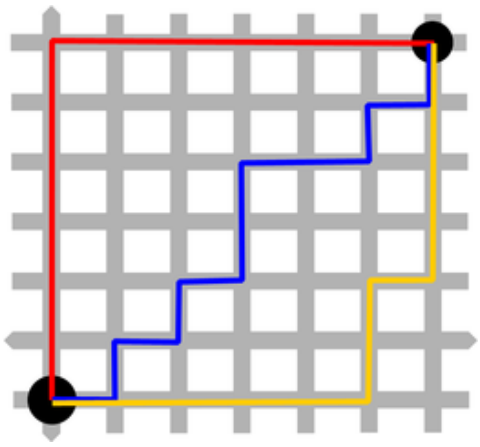
II. DASAR TEORI

A. *Jarak Manhattan*

Jarak manhattan adalah jarak dari dua titik yang diukur dengan menjumlahkan selisih absolut koordinat kartesian kedua titik. Jarak manhattan juga dikenal sebagai *rectilinear distance*, *taxicab geometry*, L_1 *distance*, *snake distance*, *city block distance*. Nama jarak manhattan berasal dari tata letak kisi dari hampir semua jalan di pulau manhattan. Prinsip dari jarak manhattan adalah setiap unit gerakan hanya boleh searah dengan absis atau searah dengan ordinat.

Jarak manhattan dapat diterapkan pada berbagai macam hal. Sebagai contoh, dalam permainan catur, jarak gerakan bidak benteng diukur menggunakan jarak manhattan. Pada beberapa game 2D, gerakan pemain biasanya dibatasi pada atas, bawah, kanan, dan kiri. Pengukuran jarak pada game tersebut dapat menggunakan jarak manhattan.

Pengukuran jarak manhattan sebenarnya terdiri dari komponen pengukuran yang lebih kecil. Dalam kasus dua dimensi, komponen pengukuran jarak tersebut adalah pengukuran untuk absis dan pengukuran untuk ordinat. Kedua komponen pengukuran tersebut bekerja secara terpisah. Sifat tersebutlah yang akan dimanfaatkan untuk menyederhanakan masalah penentuan lokasi pertemuan di makalah ini.



Gambar 1. Jarak Manhattan
Sumber: https://qph.ec.quoracdn.net/main-qimg-8d64c8344fc8364e46b9712e2c51dca4

B. Algoritma Divide and Conquer

Divide and Conquer adalah strategi militer yang banyak dipakai pada sejak dulu. Strategi tersebut dikenal dengan nama *divide ut imperes*. Ilmu komputer menggunakan nama tersebut untuk sebuah strategi algoritma karena kemiripan prinsipnya dengan strategi militer tersebut.

I. Definisi *Divide and Conquer*

Algoritma *Divide and Conquer* dapat dibagi menjadi beberapa langkah.

- *Divide*

  Membagi masalah menjadi upa-masalah yang memiliki bentuk yang sama dengan semula tapi lebih kecil. [1]

- *Conquer*

  Menyelesaikan masing-masing upa-masalah yang sudah cukup kecil dengan rekursi. [1]

- *Combine*

  Menggabungkan solusi masing-masing upa-masalah sehingga membentuk solusi masalah semula. [1]

Objek permasalahan yang dibagi adalah masukan (*input*) atau *instances* yang berukuran n: tabel (larik), matriks, eksponen, dan sebagainya, bergantung pada masalahnya. [1]

Tiap-tiap upa-masalah mempunyai karakteristik yang sama (*the same type*) dengan karakteristik masalah asal, sehingga metode *Divide and Conquer* lebih natural diungkapkan dalam skema rekursif. [1]

II. Skema Umum Algoritma *Divide and Conquer*

Jika pembagian tidak selalu menghasilkan dua upa-masalah yang berukuran sama:

```
procedure DIVIDE_and_CONQUER(input n : integer)
{ Menyelesaikan masalah dengan algoritma D-and-C.
  Masukan: masukan yang berukuran n
  Keluaran: solusi dari masalah semula
}
Deklarasi
  r, k : integer
Algoritma
  if n ≤ n0 then {ukuran masalah sudah cukup kecil}
    SOLVE upa-masalah yang berukuran n ini
  else
    Bagi menjadi r upa-masalah, masing-masing
berukuran n/k
    for masing-masing dari r upa-masalah do
      DIVIDE_and_CONQUER(n/k)
    endfor
    COMBINE solusi dari r upa-masalah menjadi
solusi masalah semula
  endif
```

Jika pembagian selalu menghasilkan dua upa-masalah yang berukuran sama:

```
procedure DIVIDE_and_CONQUER(input n : integer)
{ Menyelesaikan masalah dengan algoritma D-and-C.
  Masukan: masukan yang berukuran n
  Keluaran: solusi dari masalah semula
}
Deklarasi
  r, k : integer
Algoritma
  if n ≤ n0 then {ukuran masalah sudah cukup kecil}
    SOLVE upa-masalah yang berukuran n ini
  else
    Bagi menjadi 2 upa-masalah, masing-masing
berukuran n/2
    DIVIDE_and_CONQUER {upa-masalah pertama yang
berukuran n/2}
    DIVIDE_and_CONQUER {upa-masalah kedua yang
berukuran n/2}
    COMBINE solusi dari 2 upa-masalah
  endif
```

[1]

III. Kompleksitas Waktu Algoritma

Jika diberikan input data sebesar n, maka algoritma *Divide and Conquer* memiliki kompleksitas waktu T dengan rumus:

$$T(n) = \begin{cases} g(n), n \le n_0 \\ 2T\left(\frac{n}{2}\right) + f(n), n > n_0 \end{cases}$$

Persamaan 1. Persamaan Kompleksitas Algoritma

IV. Keunggulan Algoritma

1. Penyelesaian masalah sulit

   Jika diberikan masalah sulit, kadang kita merasa masalah tersebut terlalu sulit untuk dipecahkan karena kita tidak punya ide sedikitpun cara menyelesaikan masalah tersebut. Namun, dengan menggunakan algoritma *Divide and Conquer*, masalah tersebut dikurangi derajat kesulitannya dengan cara membaginya menjadi upa-masalah yang lebih kecil. Dengan ukuran upa-masalah yang kecil, ide untuk menyelesaikannya bisa lebih mudah didapatkan.

2. Kesangkilan Algoritma

   Paradigma *Divide and Conquer* sering berperan membantu penemuan algoritma yang efisien. Algoritma *Divide and Conquer* menjadi dasar algoritma perkalian matriks Karatsuba, *quick sort*, *merge sort*, perkalian matriks Strassen, dan pencarian median.

3. Penggunaan *Memory Cache*

   Karena algoritma ini membagi masalah menjadi upa-masalah yang lebih kecil, jika ukuran upa-masalah tersebut cukup kecil untuk bisa dimasukkan ke dalam *cache*, algoritma ini dapat berjalan lebih cepat. Hal ini dikarenakan kecepatan akses *cache* yang jauh lebih cepat daripada kecepatan akses memori.

4. Penggunaan *Multiprocessor*

   Karena algoritma ini upa-masalah yang dibagi oleh *Divide and Conquer* tidak berhubungan satu sama lain, satu penyelesaian upa-masalah dapat dilakukan di satu thread. Dengan melihat perkembangan teknologi *processor* yang mengandalkan penambahan *core*, algoritma *Divide and Conquer* akan menjadi lebih cepat.

C. *Median*

Median atau nilai tengah adalah salah satu ukuran pemusatan data. Median membagi data paruh tinggi dengan data paruh rendah. Median sering digunakan dalam pengukuran data dalam statistik dan teori probabilitas. Keunggulan median jika dibandingkan dengan rata-rata adalah median tidak tidak terlalu dipengaruhi oleh nilai data yang sangat besar atau sangat kecil. Sifat tersebut membuat median dapat memberikan ide tentang nilai tipikal dari data.

Jika ada sekumpulan data a yang terurut sebanyak n, maka median:

$$median(a) = \frac{a_{\lfloor \#\frac{n}{2} \rfloor} + a_{\lfloor \#\frac{n}{2}+0.5 \rfloor}}{2}$$

Persamaan 2. Persamaan Median

i	1	2	3	4	5	6	7	8
Data ke-i	2	2	3	6	12	13	13	30
Median adalah rata-rata data ke-4 dan data ke-5 **Median = (6 + 12) / 2 = 9**								

Tabel 1. Median dengan Jumlah Data Genap

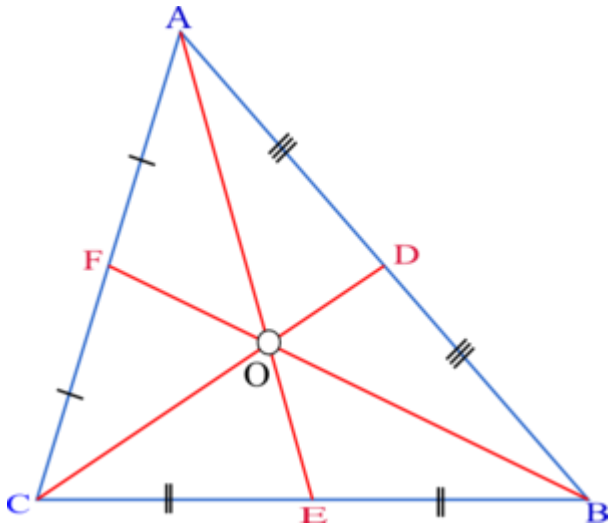
i	1	2	3	4	5	6	7	8	9
Data ke-i	2	2	3	6	12	13	13	30	35
Median adalah data ke-5 **Median = 12**									

Tabel 2. Median dengan Jumlah Data Ganjil

Median memiliki sifat meminimalkan mean dari deviasi absolut dari sekumpulan data. Sifat inilah yang dimanfaatkan untuk mendapatkan lokasi pertemuan yang optimal. Pencarian median dapat dilakukan dengan kompleksitas waktu n saja.

D. *Geometric Median*

Geometric Median dari set diskrit dari titik sampel dalam ruang Euclidean adalah titik yang meminimalkan penjumlahan dari jarak titik ke titik sampel. Geometric Median menggeneralisasikan median.



Gambar 2. Geometric Median
Sumber:
https://upload.wikimedia.org/wikipedia/commons/thumb/c/c1/Triangle.Centroid.Median.png/300px-Triangle.Centroid.Median.png

Meskipun Geometric Median secara konsep mudah dimengerti, perhitungannya sulit dilakukan. Telah dibuktikan bahwa tidak ada *closed-form expression* ataupun algoritma pasti yang menggunakan *arithmetic operations* dan *kth roots* yang bisa menyelesaikan Geometric Median.

Pencarian Geometric Median biasanya didekati dengan prosedur iteratif yang setiap langkahnya memberikan taksiran yang lebih akurat. Hal ini memanfaatkan fakta bahwa jumlah dari jarak sebuah titik ke titik sampel merupakan fungsi konveks. Fakta tersebut memberikan fakta lain yaitu fungsi tersebut hanya memiliki satu titik optimal. Salah satu algoritma yang dipakai adalah algoritma Weiszfeld.

Geometric Median akan sama dengan median jika dimensi Geometric Median sama dengan satu. Perhitungan jarak manhattan menyederhanakan Geometric median yang awalnya menggunakan dua dimensi menjadi satu dimensi.

III. IMPLEMENTASI ALGORITMA

A. *Definisi Persoalan*

Diberikan sebuah larik berisi n buah lokasi rumah dalam kisi dua dimensi dengan ukuran tak hingga. Setiap rumah memiliki posisi (x, y) yang unik.

Jarak dari dua buah rumah adalah jarak manhattan, yaitu penjumlahan dari selisih absolut x dari kedua rumah dan selisih absolut y dari kedua rumah.

Tentukan lokasi pertemuan optimal yang jumlah dari jarak setiap rumah ke lokasi pertemuan minimal!

Berikut diberikan contoh persoalan:

Larik Lokasi Rumah							
(0, 1)	(5, 6)	(1, 8)	(3, 5)	(6, 2)	(6, 3)	(7, 2)	(9, 7)

B. *Pemisahan Komponen Posisi (Divide)*

Pisahkan dari posisi tiap rumah komponen x dan komponen y ke dalam larik terpisah. Pembagian dilakukan sampai dimensi mencapai satu. Untuk kasus ini, dimensi hanya dua sehingga pembagian dapat dilakukan satu kali saja.

Pemisahan pada langkah ini hanya dapat dilakukan jika metrik jarak terdiri dari metrik pengukuran independen satu dimensi. Hal tersebut dipenuhi oleh metrik jarak manhattan.

Langkah ini merupakan bagian *divide* dari algoritma *Divide and Conquer*.

```
procedure divide(output X, Y : TabelInt)
Deklarasi
  i : integer
Algoritma
  i = 0
  for posisi rumah dari larik seluruh rumah do
    x[i] = rumah.x
    y[i] = rumah.y
    i = i + 1
  endfor
```

Maka, hasil dari langkah ini adalah:

x	0	5	1	3	6	6	7	9
y	1	6	8	5	2	3	2	7

Langkah ini memiliki kompleksitas waktu O(n).

C. *Pencarian Median dari Setiap Komponen (Solve)*

Dapatkan median dengan menggunakan algoritma partisi dari Quick Sort varian 2. Partisi menghasilkan setengah elemen senarai lebih kecil atau sama dengan pivot p dan setengah bagian lagi lebih besar dari pivot p.

Misalkan s adalah posisi pem-partisian. Jika s = ceiling(n/2), maka pivot p adalah nilai median yang dicari. Jika s > ceiling(n/2), maka median terdapat pada setengah bagian kiri. Jika s < ceiling(n/2), maka median terdapat pada setengah bagian kanan.

Langkah ini merupakan bagian *solve* dari algoritma *Divide and Conquer*.

```
procedure Partisi2(input/output A : TabelInt, input
i, j : integer, output q : integer)
{ Mengurutkan tabel A[i..j] dengan algoritma Quick
Sort.
  Masukan: Tabel A[i..j] yang sudah terdefinisi
elemen-elemennya
  Keluaran: Tabel A[i..j] yang sudah terurut menaik.
}
Deklarasi
  pivot : integer
  p : integer
Algoritma
  pivot ← A[i]
  p ← i
  q ← j + 1
  repeat
    repeat
      p ← p + 1
    until A[p] >= pivot
    repeat
      q ← q + 1
    until A[q] <= pivot
    swap(A[p], A[q])
  until p >= q
  swap(A[p], A[q])
  swap(A[i], A[q])
```

```
procedure Median(input A : TabelInt, input i, j : integer, output m : integer)
{ Mendapatkan median dari A menggunakan algoritma quicksort varian 2
  Masukan: Tabel A[i..j] yang sudah terdefinisi elemen-elemennya
  Keluaran: Median dari tabel A
}
Deklarasi
  s : integer
  k : integer
Algoritma
  s ← ceiling(Ukuran A / 2)
  Partisi2(A, i, j, k)
  while s != k do
    if k > s then
      Partisi2(A, i, k-1)
    else
      Partisi2(A, k+1, j)
  endwhile
  output(A[s], A[floor(s – 0.5)])
```

Maka, hasil dari langkah ini adalah:

Median	
x	**y**
5 – 6	3 – 5

Langkah ini memiliki kompleksitas algoritma O(n).

D. Penggabungan Irisan Median dari Tiap Komponen (Conquer)

Dapatkan irisan dari setiap median komponen. Daerah irisan yang didapatkan adalah daerah optimal yang dapat dijadikan lokasi pertemuan.

```
procedure Combine(input xMin, yMin, xMax, yMax : integer)
Deklarasi
  type Point : record < x, y : integer >
  min, max : Point
Algoritma
  min.x = xMin
  min.y = yMin
  max.x = xMax
  max.y = yMax
  output("Daerah yang optimal ada diantara", min, "dan", max)
```

Maka, hasil dari langkah ini adalah:

9										
8		R								
7										R
6						R				
5				R						
4										
3							R			
2							R	R		
1	R									
0										
y/x	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

Gambar 3. Peta Rumah dan Irisan Median

Langkah ini memiliki kompleksitas algoritma O(log n).

IV. Hasil Eksperimen dan Analisis

A. Nilai Tiap Petak

Pada Gambar 4 diperlihatkan posisi rumah dan nilai dari masing-masing petak dan irisan median.

9	75	69	65	61	59	57	57	61	67	73
8	67	61	57	53	51	49	49	53	59	65
7	64	55	54	47	45	43	43	57	53	59
6	57	51	47	43	41	39	39	43	49	55
5	55	49	45	41	39	37	37	41	47	53
4	55	59	45	41	39	37	37	41	47	53
3	55	59	45	41	39	37	37	41	47	53
2	57	51	47	43	41	39	39	43	49	55
1	63	57	53	49	47	45	45	59	55	61
0	71	65	61	57	55	53	53	57	63	69
y/x	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

Gambar 4. Peta Nilai Petak dan Irisan Median

Nilai petak adalah penjumlahan dari jarak petak tersebut ke setiap rumah. Dari Gambar 4, dapat dilihat bahwa tiap petak pada daerah irisan merupakan petak yang paling optimal untuk dijadikan lokasi pertemuan. Terlihat juga bahwa nilai petak memiliki fungsi konveks. Semakin jauh dari irisan, semakin besar pula nilai petaknya.

B. Penyempurnaan Algoritma

Algoritma ini masih bisa disempurnakan untuk menyelesaikan masalah yang memiliki batasan tambahan yang biasa dibutuhkan, yaitu lokasi pertemuan harus salah satu dari rumah yang tersedia. Untuk mendapatkan rumah tersebut, dapat didekati dengan mendapatkan beberapa rumah terdekat lalu memeriksa nilai petak rumah-rumah tersebut. Namun, pendekatan ini masih bisa memberikan solusi yang tidak optimum, namun cukup baik untuk dijadikan pendekatan.

C. Kesalahan Umum

Kesalahan yang sering dilakukan seseorang dalam memecahkan masalah ini adalah mengasumsikan bahwa lokasi yang optimal adalah lokasi rata-rata dari x dan y. Kasus yang dapat menyebabkan rata-rata tidak menghasilkan lokasi yang optimal adalah jika ada sedikit lokasi yang memiliki jarak sangat jauh dari rumah lainnya, seperti pada Tabel berikut:

Lokasi Rumah				
0	1	2	3	12

Hal tersebut terjadi jika rata-rata dari suatu data tidak sama dengan median data.

D. Kompleksitas Algoritma

Pada langkah *divide*, akan terbentuk upa-masalah sebanyak $\log_2 d$ dengan d adalah jumlah dimensi lokasi. Untuk kasus ini, dimensi yang digunakan hanya dua, dan pada umumnya, dimensi yang digunakan sangat kecil (umumnya dua dimensi atau tiga dimensi).

Pada langkah *conquer*, tiap upa-masalah akan dicari median menggunakan *Quick Sort* varian 2 sehingga kompleksitas tiap upa-masalah. Langkah ini memiliki kompleksitas algoritma O(n).

Pada langkah *solve*, tiap upa-masalah akan memberikan solusinya untuk digabungkan. Pada langkah *divide*, banyak upa-masalah yang terbentuk adalah $\log_2 d$.

Dengan demikian, kompleksitas algoritma untuk algoritma ini adalah $O(n \log d)$.

REFERENCES

[1] Munir, Rinaldi. Diktat Kuliah IF2211 Strategi Algoritma. Program Studi Teknik Informatika ITB.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 13 April 2018

Felix Septianus Darmawan 13516041