Analisis dan Penerapan Algoritma *Divide and Conquer* Dalam Penyelesaian Masalah *Convex Hulls*

Geri Noorzaman

Jurusan Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha No.10 Bandung
e-mail: if15050@students.if.itb.ac.id

ABSTRAK

Permasalahan *convex hull* adalah sebuah permasalahan yang memiliki aplikasi terapan yang cukup banyak, seperti pada permasalahan grafika komputer, otomasi desain, pengenalan pola *(pattern recognition*), dan penelitian operasi.

Banyak sekali terdapat algoritma untuk menyelesaikan permasalahan *convex hull* ini, misalnya saja dengan pendekatan *brute force*, algoritma *gift wrapping*, algoritma *Graham's Scan*, dengan metode *Divide and Conquer*, dll. Tiap algoritma memiliki kompleksitas yang berbeda-beda, hal ini tentu saja sangat bergantung pada metode atau pendekatan apa yang digunakan pada algoritma tersebut. Misalkan saja Brute Force O(n^4), Gift Wrapping O(nh), Gra - ham Scan O(n log n), Jarvis March O(nh), QuickHull O(nh), Divide-and-Conquer O(n log n), Monotone Chain O(n log n), Incremental O(n log n)

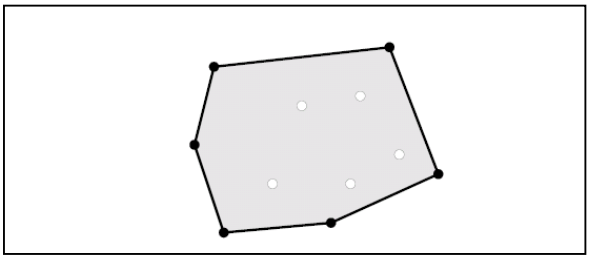
Pada makalah ini, penulis akan mencoba membahas penyelesaian permasalahan *convex hull* ini dengan metode *divide and conquer*, dan khususnya untuk *convex hull 2-D*. Pada permasalahan convex hull ini, algoritma *divide and conquer* mempunyai kompleksitas waktu yang cukup kecil, yaitu hanya O(n log n), dan selain itu juga algoritma ini memiliki beberapa kelebihan dan dapat digeneralisasi untuk permasalahan *convex hull* yang melibatkan dimensi lebih dari tiga.

Kata kunci: *convex hull, divide and conquer*.

I. PENDAHULUAN

Permasalahan dalam pencarian *convex hull*, dalam praktiknya, diaplikasikan dalam permasalahan grafika komputer, otomasi desain, pengenalan pola *(pattern recognition*), penelitian operasi, pemrosesan gambar, statistik, dll. *Convex hull* ini juga merupakan sebuah alat untuk membangun blok untuk sejumlah algoritma komputasi geometrik. Sebagai contoh, permasalahan dalam pencarian diameter dari himpunan titik-titik, yang merupakan pasangan titik-titik dengan jarak yang maksimum. Solusi dari diameternya selalu merupakan jarak antara dua buah titik dalam *convex hull*.

Pada permasalahan *convex hull*, kita diberikan himpunan titik P. Kita ingin menghitung suatu *convex hull* dari P. Secara formal, *convex hull* adalah himpunan *convex* yang mengandung P. Atau *convex hull* dapat didefinisikan juga sebagai *convex* poligon terbesar yang titik-titiknya merupakan keseluruhan titik-titik pada P. Setiap vertex dari convex hull dinamakan dengan *extreme point*, dan keluarannya berupa *list-of-extreme points*.



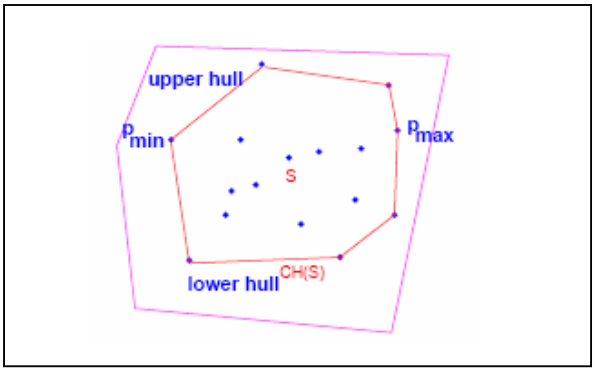
Gambar 1. Himpunan titik-titik berikut *convex hull*-nya (titik-titik[verteks*] convex hull*)

II. INSTANSIASI MASALAH

Secara sederhana, pemasalahan *convex hull* ini adalah diberikan sebuah himpunan titik-titik dalam sebuah bidang, lalu temukan *convex hull* dari titik-tititk ini dengan keluaran dapat berupa *list-of-verteks*. Atau lebih formalnya:

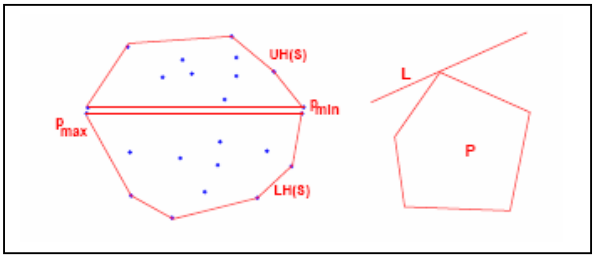
Input: Himpunan S = (p1, p2, ..., pn) dimana p1,p2,...,pn merupakan titik-titik pada bidang koordinat kartesius.

Output*: Convex Hull* CH(S) dari n titik-titik diatas.



Gambar 2. Instansiasi permasalahan dari *Convex Hull*

- Misalkan Pmax dan Pmin adalah dua buah titik pada himpunan S yang masing-masing merupakan titik maksimum dan titik minimum berdasarkan koordinat absis-X.
- Lalu Pmax dan Pmin merupakan verteks dari *convex hull*
- Lalu, jika Pmax dan Pmin dihubungkan, akan terdapat garis yang akan membagi *convex hull* menjadi dua bagian, yaitu *upper hull* dan *lower hull*.
- Garis L disebut dengan tangen dari *convex* poligon P jika semua verteks dari P berada pada sisis yang sama dari L.



Gambar 3. Contoh upper hull dan down hull serta gambaran tangen dari suatu convex P

III. PEMECAHAN MASALAH

2.1 Algoritma *Divide and Conquer*

Divide and Conquer adalah metode pemecahan masalah yang bekerja dengan membagi masalah menjadi beberapa upa-masalah yang lebih kecil, kemudian menyelesaikan masing-masing upa-masalah tersebut secara independen, dan akhirnya menggabungkan solusi masing-masing upa-masalah sehingga menjadi solusi dari masalah semula.

Pada algortima *Divide and Conquer* ini meiliki tiga proses utama yaitu:

- *Divide*: membagi masalah menjadi beberapa upa-masalah yang memiliki kemiripan dengan masalah semula namun berukuran lebih kecil (idealnya berukuran hampir sama),
- *Conquer*: memecahkan (menyelesaikan) masing-masing upa-masalah (secara rekursif), dan
- *Combine:* mengabungkan solusi masing-masing upa-masalah sehingga membentuk solusi masalah semula.

Pada algoritma ini, 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. Secara umum, algoritma *Divide and Conquer* memiliki sekema sbb:

```
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
```

2.2 Pemecahan Masalah *Convex Hull* dengan Algoritma *Divide and Conquer*

Pada penyelasaian masalah pencarian *Convex Hull* dengan menggunakan algoritma *Divide and Conquer*, hal ini dapat dipandang sebagai generalisasi dari algoritma pengurutan *merge sort*. Berikut ini merupakan garis besar gambaran dari algoritmanya:

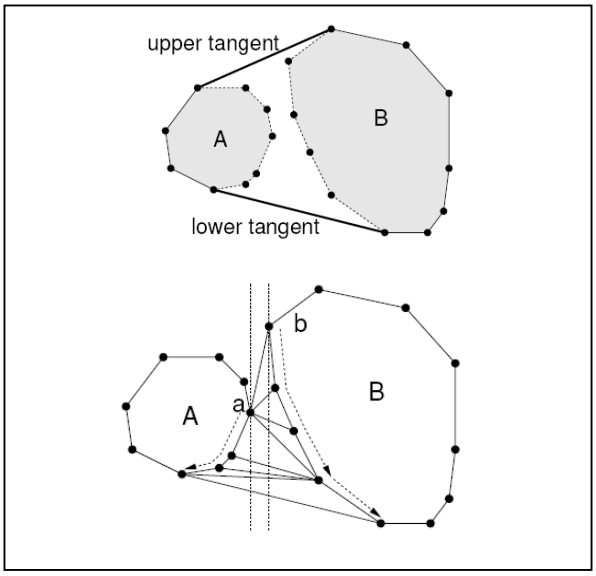
- Pertama-tama lakukan pengurutan terhadap titik-titik dari himpunan S yang diberika berdasarkan koordinat absis-X, dengan kompleksitas waktu O(n log n).

- Jika $|S| \leq 3$, maka lakukan pencarian *convex hull* secara *brute-force* dengan kompleksitas waktu O(1). (Basis)
- Jika tidak, partisi himpunan titik-titik pada S menjadi 2 buah himpunan A dan B, dimana A terdiri dari setengah jumlah dari |S| dan titik dengan koordinat absix-X yang terendah dan B terdiri dari setengah dari jumlah |S| dan titik dengan koordinat absis-X terbesar.
- Secara rekursif lakukan penghitungan terhadap H_A = conv(A) dan H_B = conv(B).
- Lakukan penggabungan (*merge*) terhadap kedua *hull* tersebut menjadi *convex hull*, H, dengan menghitung da mencari *upper* dan *lower tangents* untuk H_A dan H_B dengan mengabaikan semua titik yang berada diantara dua buah tangen ini.

Berikut ini adalah pseudo-code yang penulis buat sendiri untuk algoritmanya:

```
procedure D_and_C_CH (input P [1..n] : array
of Point, Output L : List of Point)

{ Menyelesaikan masalah convex hull dengan
algoritma D-and-C.
  Masukan: masukan array of point yang
berukuran n
  Keluaran: solusi dari masalah
}
Deklarasi
     r  : integer
     la : list of point
Algoritma

  L = {}
  if n ≤ 3₀ then  {ukuran masalah sudah cukup
kecil }
     SOLVE upa-masalah dengan metode brute-
force
  else
     Bagi menjadi r upa-masalah, masing-
masing berukuran n/k
     Ha = P[1..n/2]
     HB = P[n/2+1..n]
     C_and_D_CH(Ha)
     C_and_D_CH(HB)

     {gabungkan solusi dari r upa-masalah
menjadi solusi masalah semula }
  H = prosedur gabung Ha dan Hb dengan
      mencari lower tangen dan upper tangen
  La = listPoint(H)

  L = L ∪ la
  endif
```

Pada algoritma di atas, dapat dilihat bahwa terdapat prosedur untuk mencari lower tangen dan upper tangen. Algoritmanya merupakan suatu algoritma berjalan yang biasa. Pada kasus ini, *a* diinisialisasi sebagai titik paling kanan dari H_A dan *b* merupakan titik paling kiri dari H_B. Jika pasangan ab belum merupakan *lower tangen* untuk H_A dan H_B , maka "nailkkan" *a* menjadi *suksesor* dari *a* dan "turunkan" *b* menjadi *predesesor* dari *b*, sesuai dengan kaidah arah jarum jam.



Gambar 4. Gambaran dalam mencari lower tangen

Berikut adalah gambaran umum dalam pencarian *lower tangen* dari H_A dan H_B :

LowerTangen(H_A, H_B):

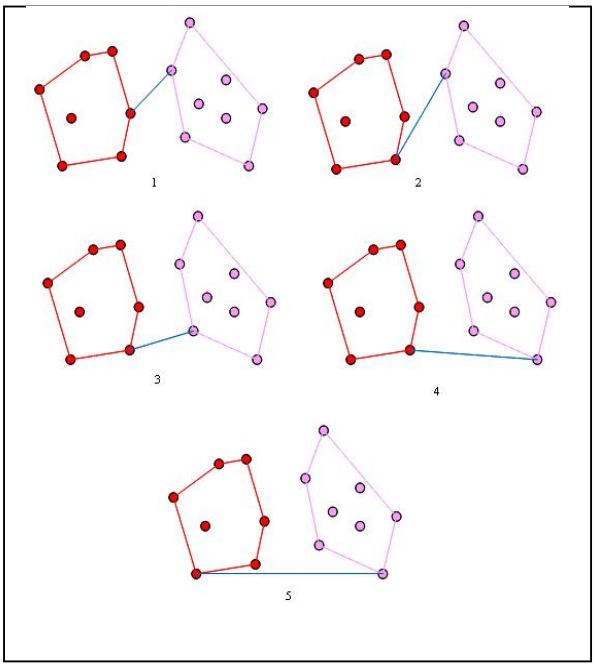
1) Misalkan *a* merupakan titik terkanan dari H_A
2) Misalkan *b* merupakan titik terkanan dari H_B
3) While (*ab* bukan merupakan lower tangen dari H_A dan H_B)do

   - While(*ab* bukan merupakan *lower tangen* dari H_A) do *a -> a.predesesor*
   - While(a*b* bukan merupakan *lower tangen* dari H_A) do *b-> b.suksesor*

4) Return *ab*

Untuk lebih jelasnya, jika dibuatkan *pseudo-code*nya akan menjadi:

```
function LowerTangent (input HA, HB : list of
point): list of point
{mencari lower tangen dari Ha dan Hb, untuk
mencari lower tangen, analogi}

Algoritma

a = rightmost point dari Ha
b = leftmost point dari Hb

while ab bukan lower tangen dar ha dan Hb do
       while ab bukan lower tangen dari Ha do
              a.pred()
       while ab bukan lower tangen dari Ha do
              b.succ()
return ab
```



Gambar 4. Gambaran lebih rinci dalam mencari lower tangen dari dua buah *hull*

Kompleksitas waktu dari algoritma yang digunakan di atas dapat dinyatakan dalam relasi rekurens. Diberikan masukkan dengan ukuran n, pertimbangkan waktu yang dibutuhkan untuk menjalankan semua bagian dari prosedur. Hal ini akan mencakup waktu untuk mempartisi himpunan titik, menghitung kedua tangen, dan mengembalikan hasil akhir. Jelas sekali bahwa langkah pertama dan ketiga dapat dijalankan dalam kompleksitas waktu O(n). Lalu dengan mengabaikan semua faktor yang konstan, algoritma di atas dapat dijelaskan dengan relasi rekurens sbb:

$$T(n) = \begin{cases} 1 & |n| < 3 \\ n + 2T(n/2) & \textit{untuk } n \textit{ lainnya} \end{cases}$$

Apabila diperhatikan, relasi rekurens tersebut sangat mirip sekali dengan relasi rekurens algoritma pengurutan MergeSort, yang jika diselesaikan akan menghasilkan kompleksitas waktu sebesar O(n log n).

IV. KESIMPULAN

Permasalahan *convex hull* adalah sebuah permasalahan yang memiliki aplikasi terapan yang cukup banyak, seperti pada permasalahan grafika komputer, otomasi desain, pengenalan pola *(pattern recognition*), dan penelitian operasi.

Divide and Conquer adalah metode pemecahan masalah yang bekerja dengan membagi masalah menjadi beberapa upa-masalah yang lebih kecil, kemudian menyelesaikan masing-masing upa-masalah tersebut secara independen, dan akhirnya menggabungkan solusi masing-masing upa-masalah sehingga menjadi solusi dari masalah semula.

Algoritma *divide and conquer* merupakan salah satu solusi dalam penyelesaian masalah *convex hull*. Algoritma ini ternyata memiliki kompleksitas waktu yang cukup kecil dan efektif dalam menyelesaikan permasalahan ini (jika dibandingkan algoritma lain). Selain itu juga, algoritma ini dapat digeneralisasi untuk permasalahan *convex hull* yang berdimensi lebih dari 3.

REFERENSI

[1] Munir, Rinaldi. 2007. *Strategi Algoritmik*. Teknik Informatika ITB : Bandung

[2] http://www.cse.unsw.edu.au/ Diakses pada tanggal 16 Mei 2007 Pukul 13.00

[3] F. P. Preparata and S. J. Hong. *Convex hulls of finite sets of points in two and three dimensions*. Commun. ACM, 20:87{93, 1977.

[2] http://softsurfer.com/Archive/algorithm_0109/algorithm_0109.htm Diakses pada tanggal 16 Mei 2007 Pukul 13.15

[3]http://www.cgal.org/Manual/3.2/doc_html/cgal_manual/Convex_hull_2_ref Diakses pada tanggal 16 Mei 2007 Pukul 13.20

[4] http://en.wikipedia.org/wiki/Convex_hull Diakses pada tanggal 16 Mei 2007 Pukul 13.30