Penerapan Algoritma Program Dinamis dalam Penjadwalan Pengerjaan Sekumpulan Tugas Pelajar

Harry Alvin Waidan Kefas - 13514036
Program Sarjana Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Bandung, Indonesia
Jalan Ganeca 10

Abstrak—Kehidupan sebagai seorang pelajar (baik siswa maupun mahasiswa) tidak akan luput dari suatu latihan di luar kegiatan tatap muka dengan pengajar yang diberikan guna memberikan waktu bagi pelajar untuk mulai/semakin mengerti dengan bagian dari ilmu yang telah diberikan oleh pengajar. Terkadang, pengajar memberikan beberapa tugas tanpa tahu berapa banyak tugas yang telah diterima oleh pelajar. Namun, sebagai pelajar, karena memang tujuan utamanya adalah mendapatkan ilmu yang ingin dimilikinya, harus lebih memanfaatkan tugas-tugas yang diberikan secara maksimal, ini berarti pelajar harus dapat melakukan penjadwalan terhadap pengerjaan tugas-tugas tersebut agar dapat diselesaikan tepat waktu dan pelajar mendapatkan juga ilmu dari tugas-tugas yang diselesaikan tersebut. Penyusunan jadwal pengerjaan tugas-tugas tersebut apabila dilakukan secara bruteforce, memakan waktu yang cukup lama. Makalah ini akan membahas mengenai cara pelajar untuk dapat menyusun jadwal pengerjaan tugas-tugas yang didapatkan agar dapat selesai sesuai *deadline* yang diberikan, dan kali ini dengan menggunakan Program Dinamis agar penjadwalan dapat diselesaikan dalam waktu yang tidak lama.

Keywords—Deadline, program dinamis, memoisasi, tahap, penjadwalan

I. PENDAHULUAN

Seringkali sebuah tugas/assignment memiliki batas waktu pengumpulan, dimana marak disebut *deadline*. Deadline merupakan sebuah momok terutama bagi para penerima deadline. Salah satu subjek yang sering mendapatkan deadline adalah pelajar. Dengan harapan pelajar dapat mengerti apa yang telah disampaikan oleh pengajar, pelajar diberikan tugas dengan deadline-deadline yang beragam oleh pengajarnya. Namun, pelajar yang merasakan sendiri bahwa tugas dengan deadline yang beragam tersebut memiliki kuantitas yang cukup membuat bingung terhadap tugas yang harus dikerjakan terlebih dahulu, dengan memberikan prioritas terhadap setiap tugas berdasar kepada deadline dari masing-masing tugas tersebut.

Khusus dari pandangan pelajar, deadline dari tugas-tugas sering dianggap sepele sehingga pada praktiknya sendiri, tugas-tugas baru dikerjakan dalam durasi pengerjaan hingga deadline dari tugas yang terbilang singkat. Maka, untuk memperbaiki keteledoran dari setiap pelajar, harus ada cara untuk melakukan penyusunan agenda pengerjaan tugas yang harapannya akan berdampak di ketuntasan tugas beberapa saat sebelum deadline.

Dalam menyelesaikan suatu permasalahan secara terstruktur dan berulang, dapat digunakan rantai prosedur yang dinamakan algoritma. Terdapat bermacam-macam algoritma dengan kelebihan dan kekurangannya masing-masing. Beberapa algoritma dapat digunakan untuk menyelesaikan satu masalah (yang berarti dalam menyelesaikan suatu masalah, dapat digunakan lebih dari satu algoritma). Penyusunan jadwal pengerjaan tugas-tugas milik pelajar dapat menggunakan algoritma. Terdapat berbagai macam algoritma untuk menyelesaikan masalah ini, seperti algoritma bruteforce, algoritma greedy, atau algoritma exhaustive search yang merupakan variasi dari algoritma bruteforce. Selain algoritma-algoritma yang telah disebutkan sebelumnya untuk menyelesaikan permasalahan ini, ada juga algoritma lainnya, dimana akan dibahas di dalam makalah ini, yaitu Algoritma Program Dinamis.

Pada umumnya, untuk persoalan-persoalan yang kompleks seperti Travelling Salesman Problem, Knapsack Problem, dapat diselesaikan dengan algoritma bruteforce, greedy, ataupun algoritma exhaustive search. Namun, waktu penyelesaian akan menjadi cukup lama untuk data yang sudah mulai besar, yang disebabkan oleh kompleksitas dari algoritma-algoritma tersebut untuk menyelesaikan permasalahan-permasalahan seperti yang diatas tidak termasuk ke dalam kompleksitas yang polinomial. Dengan menggunakan program dinamis, kompleksitas waktu dari penyelesaian persoalan-persoalan yang cukup lama diselesaikan dengan algoritma bruteforce, greedy, atau exhaustive search ini dapat selesai dalam waktu polinomial karena beberapa kompleksitas waktu dari algoritma program dinamis untuk menyelesaikan persoalan-persoalan sulit seperti diatas adalah polinomial.[1]

II. PENJADWALAN DENGAN DEADLINE[3]

A. Konsep dasar

Penjadwalan dengan deadline pada dasarnya adalah melakukan penyusunan urutan *job* yang dilakukan dengan waktu start dan durasi pengerjaan *job* masing-masing,

disamping itu juga mempertimbangkan/memperhitungkan deadline dari masing-masing *job*. Dengan melakukan penyusunan, apabila hasil penyusunan itu menghasilkan susunan yang cocok, hasil tersebut akan berdampak kepada profit yang didapatkan setelah sekuens *job* telah diselesaikan. Namun, untuk mendapatkan sekuens *job* secara mangkil terkadang membutuhkan waktu yang tidak sedikit.

B. Penjadwalan pengerjaan tugas dengan deadline

Di dalam kehidupan keseharian seorang pelajar, penjadwalan pengerjaan tugas dengan deadline yang dimiliki oleh masing-masing tugas biasanya akan membutuhkan waktu pengerjaan yang sama untuk setiap tugas, dengan mengambil rata-rata dari setiap tugas diselesaikan dalam waktu normal, yaitu 1 minggu.

Berdasarkan yang telah tertulis di referensi [1], persoalan penjadwalan pengerjaan tugas dengan deadline tidak dijelaskan. Persoalan yang dijelaskan hanya *Weighted Interval Scheduling*, persoalan penjadwalan dengan bobot tertentu dan yang dicari dalam masalah tersebut adalah pengambilan *job* dengan sekuens seperti apa yang menghasilkan profit tertinggi. Berbeda konteks dengan permasalahan penjadwalan pengerjaan tugas dengan deadline, tetapi di dalam makalah ini akan dilakukan penjadwalan pengerjaan tugas dengan deadline tersebut dengan menggunakan algoritma program dinamis serta melakukannya dengan melakukan pendekatan seperti persoalan penjadwalan yang dijelaskan pada referensi [1], sehingga dampaknya semua tugas terdapat dalam hasil yang didapatkan, dengan kata lain semua tugas terkerjakan.

III. PROGRAM DINAMIS[1][2]

Program dinamis memiliki beberapa hal yang perlu diperhatikan sebelum melakukan percobaan terhadap algoritma ini dalam penyelesaian persoalan penjadwalan pengerjaan tugas dengan deadline-deadline tertentu.

A. Konsep dasar

Beberapa persoalan dapat diselesaikan dalam waktu polinomial dengan menggunakan algoritma bruteforce. Namun, tidak sedikit persoalan yang tidak dapat diselesaikan dalam waktu polinomial apabila menggunakan algoritma bruteforce. Kompleksitas dari algoritma bruteforce terkadang selalu besar dibandingkan dengan algoritma-algoritma lainnya, karena bruteforce (dan juga variasinya, *exhaustive search*) menggunakan teknik mencoba semua kemungkinan yang dapat dikatakan sebagai solusi dari permasalahn dan pasti akan mendapatkan solusi yang paling optimal, tetapi tidak dengan waktu yang efektif.

Algoritma program dinamis memiliki metode yang hampir sama dengan metode milik bruteforce atau sejenisnya. Akan tetapi, program dinamis tidak senaif bruteforce. Program dinamis pada dasarnya akan menyimpan semua hal (dapat berupa rumus/langkah yang menghasilkan nilai) yang telah dikomputasi sebelumnya, sehingga setiap kali melakukan komputasi program dinamis akan mengecek terlebih dahulu apakah rumus/langkah yang menghasilkan nilai tersebut sudah pernah disimpan sebelumnya (dalam artian sudah pernah dikomputasi sebelumnya). Apabila hal tersebut sudah pernah dikomputasi, waktu untuk mengkomputasi hal yang sama untuk seterusnya setelah komputasi hal tersebut pertama kali hanya akan membutuhkan waktu konstan. Akibatnya, waktu untuk megnkomputasi semua kemungkinan seperti yang bruteforce lakukan (namun dalam waktu yang lamanya hingga eksponensial) dapat dilakukan dalam waktu polinomial.[2]

Algoritma program dinamis seringkali dibutuhkan dalam komputasi upapersoalan dari persoalan yang hendak dicari solusinya itu, karena apabila suatu upapersoalan yang sama dilakukan secara berulang, waktu untuk menyelesaikan persoalan akan cukup lama, sedangkan dengan program dinamis hanya dibutuhkan waktu polinomial. Hal ini dapat terjadi karena prinsip dasar yang dilakukan oleh program dinamis, yaitu memoisasi/pencatatan. Pencatatan dilakukan terhadap setiap hasil komputasi pertama kali suatu upapersoalan.[2]

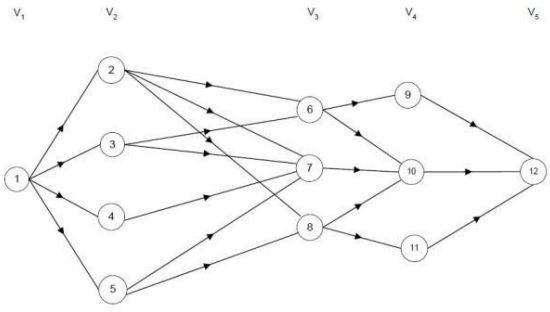
Selain konsep dari memoisasi, konsep dasar lainnya dari program dinamis adalah menggunakan pembagian persoalan menjadi beberapa upapersoalan, kemudian setiap upapersoalan akan dikomputasi lalu di akhir solusi dari setiap solusi lokal (milik upapersoalan) menjadi solusi global persoalan, dimana dengan menggunakan prinsip memoisasi, akan ada beberapa upapersoalan yang akan diselesaikan dalam waktu konstan, karena sebelumnya upapersoalan tersebut pernah dikomputasi oleh upapersoalan lainnya yang akan memengaruhi kompleksitas waktu dari algoritma menjadi polinomial.[3]

B. Karakteristik persoalan program dinamis[3]

Secara ringkas, karakteristik pada persoalan yang hendak diselesaikan dengan menerapkan program dinamis adalah seperti berikut.

1. Persoalan dapat dibagi menjadi beberapa tahap (*stage*), yang pada setiap tahap hanya diambil satu keputusan.

2. Masing-masing tahap terdiri dari sejumlah status (*state*) yang berhubungan dengan tahap tersebut. Secara umum, status merupakan bermacam kemungkinan masukan yang ada pada tahap tesebut. Jumlah status bias berhingga (*finit*e) atau tak berhingga (*infinite*). Terlihat pada gambar di bawah perbendaan antara tahap dan status yang diberikan pada graf multitahap (*multistage graph*). Tiap simpul di dalam graf tersebut menyatakan status, sedangkan V_1, V_2, .., dan seterusnya menyatakan tahap.



Gambar 1. Graf multitahap[3]

3. Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya. Artinya, setiap hasil keputusan yang diambil pada setiap tahap dimemoisasi

untuk digunakan pada tahap selanjutnya seperti konsep dasar program dinamis yang telah dipaparkan pada bagian sebelumnya.

4. Ongkos (*cost*) pada suatu tahap meningkat secara teratur (*steadily*) dengan bertambahnya jumlah tahapan. Hal ini dapat dimengerti dengan mudah karena setiap upapersoalan memiliki ongkosnya masing-masing dan saat solusi dari upapersoalan diintegrasikan kembali, ongkos global persoalan akan bertambah seiring berkurangnya upapersoalan yang tersisa.

5. Ongkos pada suatu tahap bergantung pada ongkos tahap-tahap yang sudah berjalan dan ongkos pada tahap tersebut. Hal ini cukup jelas dengan penjelasan yang sama dengan poin 4.

6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan yang dilakukan di tahap sebelumnya. Hal ini disebabkan oleh karena setiap upapersoalan (tahap) memiliki hasil optimal yang berbeda dengan upapersoalan lainnya.

7. Adanya hubungan rekursif yang mengidentifikasikan keputusan terbaik untuk setiap status pada tahap k memberikan keputusan terbaik untuk setiap status pada tahap k+1. Karakteristik ini pun merupakan salah satu cara agar program dinamis dapat melakukan memoisasi terhadap nilai/hasil dari suatu rumus/rekurens/komputasi yang sudah pernah dilakukan oleh program dinamis agar dapat digunakan kembali pada komputasi selanjutnya yang memiliki parameter yang sama dengan parameter nilai komputasi yang sudah dicatat oleh algoritma

8. Prinsip optimalitas berlaku pada persoalan tersebut. Hal ini harus diutamakan karena tujuan dari program dinamis salah satunya adalah agar permasalahan kompleks dapat diancangi dengan algoritma dinamis agar pencarian solusi dari permasalahan dapat diselesaikan dengan waktu yang tidak banyak.

Dalam menyelesaikan persoalan dengan program dinamis, dapat digunakan dua ancangan (*approach*) yang berbeda, yaitu maju (*up-down*) atau mundur (*bottom-up*). Misalkan x_1, x_2, …, x_n menyatakan peubah keputusan (hal yang menentukan parameter, indikasi bagian dari solusi) yang harus dibuat masing-masing untuk tahap 1, 2, ..., n. Maka,

a. Program dinamis maju, program dinamis akan bergerak mulai dari tahap ke-1, tahap ke-2, hingga tahap ke-n.

b. Program dinamis mundur, program dinamis akan bergerak mundur mulai dari tahap ke-n, tahap ke-(n-1), hingga tahap ke-1.

Secara umum, ada empat langkah yang dilakukan dalam mengembangkan algoritma program dinamis, yaitu

1. Karakteristikkan struktur solusi optimal (program dinamis maju atau mundur).

2. Definisikan secara rekursif nilai dari solusi optimal (membuat rumus/komputasi yang direpresentasikan rekursif untuk mendapatkan nilai optimal global dan lokal).

3. Hitung nilai solusi optimal secara maju atau mundur (lalu mencatat hasil perhitungannya).

4. Konstruksi solusi optimal (mengkonstruksikan solusi optimal lokal menjadi solusi optimal global).

IV. Tahap Penyusunan Jadwal Pengerjaan Tugas dengan Deadline

Dengan menggunakan langkah pengembangan algoritma program dinamis yang telah dipaparkan pada bagian sebelumnya, kita akan mencari solusi optimal pengerjaan tugas yang telah dimiliki dengan deadlinenya masing-masing

A. Mengarakteristikan Struktur Solusi Optimal

Algoritma program dinamis dapat menggunakan dua jenis ancangan seperti yang telah dijelaskan sebelumnya dengan menggunakan ancangan program dinamis maju yang akan melakukan tahapan dari status awal hingga status akhir atau ancangan program dinamis mundur yang akan melakukan tahapan dari status akhir hingga status awal.

B. Pendefinisian secara rekursif nilai solusi optimal

Algoritma program dinamis terkadang digunakan untuk menyelesaikan persoalan yang memiliki persamaan rekursif agar persoalan tersebut yang tadinya harus diselesaikan dengan waktu hingga eksponensial, dapat diselesaikan dengan waktu polinomial. Untuk kasus penyusunan jadwal pengerjaan tugas dengan deadline ini, definisi rekursif dari penyelesaian persoalan untuk mendapatkan runutan tugas yang harus dikerjakan sebanyak mungkin adalah sebagai berikut.

Basis :

$$M[j] = 0, j = 0$$

Rekurens :

$$M[j] = \max(V_j + M[p(j)],$$

$$M[j-1]), \text{ yang lainnya.}$$

Dimana M[j] menyatakan larik dari nilai-nilai profit maksimum yang didapatkan dari *job* ke-j apabila setelah pengambilan beberapa *job*, berakhir di *job* ke-j[1], dan untuk *job* ke-0 bobotnya adalah 0 karena *job* pertama adalah *job* ke-1. Hal ini dilakukan untuk menghindari pengaksesan indeks yang kurang dari 1 di dalam rekurens ditolak. Lalu, V_j menyatakan bobot/profit murni dari *job* ke-j. V_j digunakan sebagai tolok ukur dari keoptimalan solusi-solusi yang terdapat di dalam larik M[j]. Peubah selanjutnya adalah p(j) yang menjadi indeks dari larik M saat nilainya dijumlahkan dengan V_j. p(j) menyatakan indeks *job* yang telah dilewati sebelum menuju *job* ke-j untuk mendapatkan bobot optimal pada M[j].

Program dinamis akan mengkomputasi M[j] secara rekursif, mencari nilai maksimum dengan mencoba semua kemungkinan yang ada untuk mencapai suatu *job* ke-j, dimana rekursifnya dimemoisasi agar untuk pemanggilan M[j] di M[j+c] tidak haruss melakukan rekursif terhadap M[j], dimana $c \geq 1$.

Dalam menentukan M[j], dilakukan perbandingan bobot antara bobot dari M[j-1] dengan bobot V_j + M[p(j)]. Nilai bobot yang lebih optimal akan di-*assign* sebagai nilai M[j]. Nilai bobot yang dikatakan optimal lokal adalah nilai masing-masing M[j],

sementara nilai bobot yang dikatakan optimal global adalah nilai M[j] di larik M yang paling besar.

C. Hitung nilai solusi optimal secara maju atau mundur

Pada perhitungan nilai solusi optimal dari larik M untuk persoalan ini tidak mengadakan bobot khusus untuk setiap tugas yang diberikan deadline. Bobot yang dimiliki setiap tugas adalah deadline dari tugas itu sendiri. Deadline yang diberikan adalah durasi maksimal pengumpulan tugas dari waktu dimana tugas itu diberikan. Perhitungan ini membutuhkan tabel *hash* dari masing-masing tugas yang akan mengindikasikan tugas ke-j untuk mencegah terjadinya anomali algoritma tak hingga (*infinite algorithm*) yang akan menyebabkan tidak terselesaikannya pencarian solusi karena adanya pengaksesan sebuah sel dalam larik M yang belum ada nilainya secara rekursif dan terus menerus.

Pembuatan tabel *hash* dari setiap tugas yang diberikan ini mengurutkan terlebih dahulu waktu finish sehingga akan terjadi

$$f_1 \leq f_2 \leq f_3 \ldots \leq f_n$$

Dimana f adalah waktu selesai dari tugas tersebut. Selain pembuatan tabel *hash* dari setiap tugas, dibutuhkan juga tabel p(j), dimana range dari p(j) berbeda dengan yang dijabarkan pada [1], yaitu

$$p(j) = \max_{start(k) \leq start(j)} (deadline(k), deadline(j))$$

Mengartikan nilai indeks *hash* tugas yang nilai deadlinenya paling besar dimana waktu start dari suatu tugas ke-k lebih kecil atau sama dengan waktu start dari tugas ke-j.

D. Konstruksi solusi optimal

Setelah semua upapersoalan/tahapan dari program algoritma sudah didapatkan, setiap bagian dari upapersoalan digabungkan. Titik awal menggabungkan antara upapersoalan satu dengan upapersoalan lainnya adalah dengan melakukan tracking *hash* dari nilai yang paling optimal yang ada di dalam tabel hasil perhitungan.

Bentuk konkret dari tracking *hash* tersebut adalah dengan mengambil nilai yang paling optimal di dalam larik M yang kemudian di larik tersebut diambil indeksnya untuk mengetahui indeks sebelum indeks penampung nilai paling optimal dari larik tersebut. Indeks sebelum indeks penampung nilai optimal global itu didapatkan melalui larik P yang berisi indeks sebelum dari indeks pada larik P.

V. Penerapan Algoritma Program Dinamis dalam Melakukan Penjadwalan Pengerjaan Tugas

Algoritma program dinamis ini membutuhkan data untuk melakukan pencarian solusi runutan terbaik pengerjaan tugas-tugas yang diberikan dengan deadline yang berbeda-beda. Oleh karena itu, sebagai contoh penerapan algoritma dinamis, akan digunakan deadline-deadline dari tugas-tugas di mata kuliah IF2211 Strategi Algoritma di Institut Teknologi Bandung.

A. Kondisi Awal

Diketahui bahwa tugas-tugas yang diberikan di mata kuliah ini beserta durasi dari deadlinenya () dan waktu mulai diberikannya tugas-tugas adalah sebagai berikut.

Tugas ke-	Deadline (minggu)	Mulai diberikan di minggu ke-
1	1	1
2	2	2
3	1	2
4	3	2
5	2	3
6	3	3
7	2	4

Tabel 1. Tugas-tugas dari suatu mata kuliah dengan deadlinenya

Dengan data diatas, akan dilakukan tahapan-tahapan program dinamis untuk menyelesaikan urutan pengerjaan tugas-tugas tersebut berdasarkan pada waktu mulai diberikannya tugas-tugas tersebut dan durasi maksimal dari pemberian tugas hingga pengumpulan tugas yang bersangkutan.

B. Mengarakteristikan Struktur Solusi Optimal

Algoritma program dinamis dapat menggunakan dua jenis ancangan, yaitu program dinamis maju dan program dinamis mundur. Untuk penyelesaian contoh persoalan penjadwalan dengan deadline di upabab ini, akan diguankan ancangan program dinamis maju. Sehingga tahap akan dimulai dari tahap pertama yaitu dengan menggunakan basis-rekurens dari upabab sebelumnya di bagian C, pengisian larik M dilakukan dari indeks yang paling awal (indeks ke-0).

C. Pendefinisian secara rekursif nilai solusi optimal

Sebelum melakukan pendefinisian secara rekursif untuk mendapatkan nilai solusi optimal, terlebih dahulu dibuat hash dari setiap tugas ke-j untuk dapat digunakan dalam pendefinisian secara rekursif itu. Hasil pembuatan hash berdasarkan urutan deadline dari setiap tugas yang ada di dalam persoalan adalah sebagai berikut.

Tugas ke-	*Hash* (indeks)
1	1
2	3
3	2
4	4
5	5
6	6
7	7

Tabel 2. Hasil pembuatan *hash* dengan melakukan sorting deadline tugas terlebih dahulu

Setelah mendapatkan *hash* dari masing-masing tugas, setiap *hash* dari tugas diperlakukan definisi rekursif berikut.

Basis :

$$M[j] = 0, j = 0$$

Rekurens :

$$M[j] = \max(V_j + M[p(j)],$$

$$M[j-1]), \text{ yang lainnya.}$$

Dengan p(j) ditentukan melalui kolom berikut.

Tugas ke-(*hash*)	1	2	3	4	5	6	7
p(j)	0	2	1	3	4	5	6

Tabel 3. Hasil komputasi persamaan p(j)

Dimana nilai p(j) didapatkan dari

$$p(j) = \max_{start(k) \leq start(j)}(deadline(k), deadline(j))$$

D. Hitung nilai solusi optimal secara maju

Untuk menemukan solusi optimal lokal persoalan dan menyimpannya ke dalam larik M dengan menggunakan prinsip program dinamis maju, berikut algoritmanya[1].

```
Masukan : n, S1..Sn, F1..Fn, V1..Vn
Mengurutkan tugas sesuai finish timenya sehingga F1 ≤ F2 ≤ .. ≤ Fn
Mengkomputasi p(1), p(2), …, p(n)
for j = 1 to n
  M[j] = empty
endfor
M[0] = 0

KomputasiM(j) : void
  if (M[j] is empty) then
     M[j] = max(deadline(j) + M[p(j)],
                M[j-1])
  endif
  return M[j]
```

Algoritma diatas akan diterapkan kepada larik M. Berikut upapersoalan-upapersoalan (tahap-tahap) yang dikomputasi hingga mendapatkan optimal global dari masing-masing upapersoalan.

Tahap Awal :

M[j]	0							
p(j)	-	0	1	2	3	4	5	6
Hash	-	1	2	3	4	5	6	7

Tahap 1 :

M[j]	0	Max(0+1,0) = 1						
p(j)	-	0	1	2	3	4	5	6
Hash	-	1	2	3	4	5	6	7

Tahap 2 :

M[j]	0	1	Max(1+1,1) = 2					
p(j)	-	0	1	2	3	4	5	6
Hash	-	1	2	3	4	5	6	7

Tahap 3 :

M[j]	0	1	2	Max(2+2,2) = 4				
p(j)	-	0	1	2	3	4	5	6
Hash	-	1	2	3	4	5	6	7

Tahap 4 :

M[j]	0	1	2	4	Max(4+3,4) = 7			
p(j)	-	0	1	2	3	4	5	6
Hash	-	1	2	3	4	5	6	7

Tahap 5 :

M[j]	0	1	2	4	7	Max(7+2,7) = 9		
p(j)	-	0	1	2	3	4	5	6
Hash	-	1	2	3	4	5	6	7

Tahap 6 :

M[j]	0	1	2	4	7	9	Max(9+3,9) = 12	
p(j)	-	0	1	2	3	4	5	6
Hash	-	1	2	3	4	5	6	7

Tahap 7 :

M[j]	0	1	2	4	7	9	12	Max(12+2,12) = 14
p(j)	-	0	1	2	3	4	5	6
Hash	-	1	2	3	4	5	6	7

E. Konstruksi solusi optimal

Setelah semua upapersoalan/tahapan dari program algoritma sudah didapatkan, setiap bagian dari upapersoalan digabungkan.

Didapatkan bahwa solusi paling optimum yang ada di dalam larik M adalah 14 yang berada pada indeks ke-7 dari larik M. Sehingga, *hash* yang terbentuk dari hasil optimal tersebut adalah sebagai berikut.

$$7 \leftarrow 6 \leftarrow 5 \leftarrow 4 \leftarrow 3 \leftarrow 2 \leftarrow 1$$

Diketahui sebelumnya bahwa *hash* hanya digunakan agar proses penyusunan urutan tugas yang harus dikerjakan dengan program dinamis dapat dimengerti dengan menggunakan algoritma dalam makalah ini. Maka, *hash* harus diubah kembali ke makna asli dari masing-masing *hash* tersebut. Sehingga solusi optimal global dari persoalan yang diberikan pada upabab ini adalah sebagai berikut.

$$7 \leftarrow 6 \leftarrow 5 \leftarrow 4 \leftarrow 2 \leftarrow 3 \leftarrow 1$$

Jadi, urutan pengerjaan tugas yang memungkinkan hanya satu pada kasus ini, yaitu mengerjakan tugas 1 terlebih dahulu, kemudian dilanjutkan dengan tugas 2, tugas 3, tugas 4, tugas 5, tugas 6, lalu tugas 7.

F. Kompleksitas

Algoritma program dinamis untuk menyelesaikan persoalan ini memiliki kompleksitas waktu O(n) dimana dalam komputasi p(j) adalah O(n) serta kompleksitas waktu untuk melakukan kalkulasi M[j] dan menyimpannya ke dalam larik adalah konstan atau O(1), sehingga kompleksitas dari program dinamis ini dalam menyelesaikan persoalan penjadwalan ini adalah O(n).

KESIMPULAN

Algoritma program dinamis merupakan salah satu algoritma yang dapat membantu penyelesaian persoalan keputusan. Salah satu persoalan keputusan yang dapat diselesaikan dengan menggunakan program dinamis adalah persoalan yang dialami oleh beberapa pelajar, yaitu penjadwalan pengerjaan tugas yang memiliki deadline. Persoalan ini dapat diselesaikan dengan program dinamis dengan kompleksitas yang jauh berbeda dengan algoritma *greedy*, yaitu O(n). Meskipun penyelesaian persoalan ini memiliki kompleksitas waktu yang lebih baik, akan tetapi sampai saat ini belum ada algoritma yang cukup mangkus untuk menyelesaikan persoalan penjadwalan ini, karena persoalan penjadwalan ini merupakan NP-complete, sehingga jika ada algoritma yang dapat menyelesaikan persoalan NP-complete dalam waktu polinomial, maka persoalan penjadwalan juga dapat diselesaikan dalam waktu polinomial.

UCAPAN TERIMA KASIH

Pertama-tama penulis ingin mengucapkan syukur kepada Tuhan Yang Maha Esa karena oleh rahmat-Nya penulis dapat menyelesaikan makalah ini. Penulis juga ingin berterima kasih kepada Ibu Nur Ulfa Maulidevi dan Bapak Rinaldi Munir yang telah mengajarkan mata kuliah Strategi Algoritma di semester dan tahun ajaran ini, termasuk mengajarkan materi Program Dinamis, sehingga penulis mampu membuat makalah ini. Disamping itu, penulis juga ingin mengucapkan terima kasih kepada orang tua dan rekan-rekan yang telah memberikan semangat dan dorongan kepada penulis untuk terus belajar.

DAFTAR PUSTAKA

[1] https://www.cs.princeton.edu/courses/archive/spr05/cos423/lectures/06dynamic-programming.pdf, diakses pada 7 Mei 2016.

[2] https://youtu.be/OQ5jsbhAv_M, diakses pada 3 Mei 2016.

[3] Munir, Rinaldi. Diktat Kuliah IF2211 Strategi Algoritma. Program Studi Teknik Informatika. Institut Teknologi Bandung. 2009.

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, 8 Mei 2016

Harry Alvin Waidan Kefas (13514036)