Aplikasi Resource Management di Perusahaan Berbasis Proyek dengan Algoritma Dynamic Programming

Michael Jonathan Halim - 13521124
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail : 13521124@std.stei.itb.ac.id

***Abstract*—Pengalokasian sumber daya pada perusahaan bukanlah suatu hal yang mudah bagi pekerja yang awam dengan metode-metode komputasi. Penelitian ini bertujuan untuk membuat algoritma optimisasi pengalokasian sumber daya di dalam perusahaan berbasis proyek menggunakan *dynamic programming*. Dengan memanfaatkan algoritma tersebut, akan dirancang sebuah aplikasi website agar implementasi dapat digunakan dalam kasus yang nyata untuk membantu mengembangkan perusahaan.**

Keywords—sumber daya; komputasi; dynamic programming; website;

I. PENDAHULUAN

Banyak perusahaan atau start-up yang berbasis proyek membutuhkan suatu pengelolaan alokasi berbagai sumber daya yang dimiliki untuk berbagai proyek yang dilayani. Pengelolaan ini dilakukan oleh pemilik perusahaan ataupun karyawan lainnya yang ditugaskan untuk melakukan pengelolaan tersebut. Namun, terdapat faktor utama yang membuat pengelolaan menjadi tidak semudah yang dipikir. Faktor tersebut adalah sumber daya perusahaan yang terbatas dan tentunya suatu proyek memiliki requirement sumber daya minimum untuk dilaksanakan.

Suatu proyek biasanya memiliki kebutuhan sumber daya minimal dan penghasilan yang didapatkan jika proyek berhasil dilaksanakan. Akibatnya, tidak semua proyek yang ditawarkan dapat diambil oleh perusahaan. Oleh karena itu, tentu seorang CEO perusahaan ingin mendapatkan keuntungan yang sebesar-besarnya untuk kemajuan perusahaannya juga.

Dalam rangka mencapai keuntungan yang sebesar-besarnya, CEO perusahaan perlu melakukan analisis yang tepat dan menentukan strategi khusus untuk menentukan alokasi sumber daya yang optimal. Tentu untuk jumlah proyek yang semakin besar diperlukan penggunaan alat bantu pengambil keputusan yang menggunakan suatu algoritma khusus untuk menciptakan solusi yang optimal. Dengan melakukan pengelolaan alokasi sumber daya yang efisien, perusahaan dapat meningkatkan produktivitas dan mencapai kemajuan juga dalam bisnis mereka.

Gambar 1.1 Manajemen Pengelolaan Sumber Daya Perusahaan

Sumber: Google Images

Untuk mendapatkan keuntungan yang terbaik, dapat dimanfaatkan dynamic programming untuk menentukan alokasi sumber daya kepada proyek-proyek yang dapat dilaksanakan.

II. LANDASAN TEORI

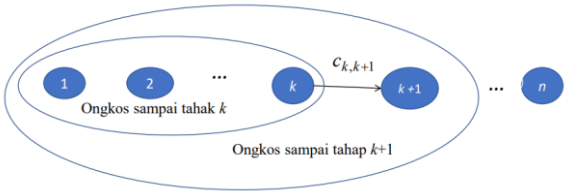
A. *Dynamic Programming*

Program dinamis (*dynamic programming*) merupakan metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan tahapan/*stage* sehingga solusi persoalan dapat dipandang sebagai serangkaian keputusan yang saling berkaitan. Kata "program" disini tidak ada katiannya sama sekali dengan pemrograman dan istilah "dinamis" muncul karena dalam pencarian solusi, dilakukan perhitungan dengan menggunakan tabel yang dapat berkembang seiring pengisian.

Program dinamis digunakan untuk menyelesaikan persoalan optimisasi (dalam kasus ini adalah optimisasi pengelolaan sumber daya untuk menghasilkan keuntungan terbesar). Namun, walaupun *greedy* juga menyelesaikan persoalan optimisasi, terdapat perbedaan yang signifikan di antara keduanya tersebut. Greedy hanya menghasilkan satu rangkaian keputusan saja sedangkan program dinamis

menghasilkan lebih dari satu rangkaian keputusan yang dipertimbangkan.

Terdapat prinsip optimalitas yang diterapkan pada program dinamis untuk membuat rangkaian keputusan yang optimal. Prinsip itu menjelaskan bahwa jika solusi total optimal, maka bagian sampai tahap ke-*k* juga optimal sehingga jika kita bekerja dari tahap *k* ke tahap *k*+1, kita tidak perlu menghitung kembali dari awal karena pada tahap *k*, hasil sudah optimal.



Gambar 2.1 Prinsip Optimalitas

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf

Persoalan yang dapat diselesaikan dengan program dinamis memiliki tujuh karakterisik utama, yaitu

1. Persoalan dapat dibagi menjadi beberapa *stage* dan pada setiap *stage* hanya diambil satu keputusan
2. Masing-masing tahap terdiri dari sejumlah *state* yang berhubungan dengan *stage* tersebut. *State* disini merupakan kemungkinan-kemungkinan masukan yang ada pada suatu *stage*. Karakteristik ini dapat diilustrasikan dengan graf multitahap.
3. Hasil dari keputusan yang diambil pada setiap *stage* ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya.
4. Ongkos pada suatu *stage* meningkat secara teratur sesuai bertambahnya jumlah tahapan.
5. Ongkos pada suatu tahap bergantung pada ongkos tahap-tahap yang sudah berjalan dan ongkos dari tahap tersebut ke tahap berikutnya.
6. Adanya rekursif yang mengambil keputusan terbaik untuk setiap status pada tahap *k* sehingga keputusan tersebut dapat digunakan untuk tahap *k*+1.
7. Prinsip optimalitas berlaku pada persoalan tersebut.

Terdapat dua pendekatan pada program dinamis, yaitu

1. *Top-down*

   Pendekatan dengan *top-down* merupakan pendekatan yang dilakukan dengan perhitungan dilakukan dari tahap 1 hingga *n*. Dengan pendekatan *top-down*, kita mulai dengan masalah utama dan secara rekursif mempertimbangkan bergerak ke bawah yaitu ke tahap-tahapnya untuk menyelesaikan masalah keseluruhan. Teknik yang digunakan dalam *top-down* disebut dengan *memoization* (*cache* penyimpanan hasil fungsi sementara), untuk menyimpan data-data sebelumnya yang berkemungkinan untuk digunakan untuk langkah selanjutnya. Karakteristik utama dari *top-down* adalah rekursif dan *memoization*.

Fibonacci Memoization Hash Table

key (function input)	0	1	2	3	4	5	6	7
value (function result)	0	1	1	2	3	5	8	13

Gambar 2.2 Contoh Pengunaan *Memoization* DP

Sumber: Google Images

2. *Bottom-up*

   Pendekatan dengan *bottom-up* merupakan pendekatan yang dilakukan dengan perhitungan dilakukan dari tahap ke *n* hingga tahap ke-1. Dengan pendekatan *bottom-up*, kita mulai dengan permasalahan kecil ke permasalahan yang lebih besar. Teknik utama yang digunakan dalam *bottom-up* disebut dengan *tabulation*, dimana kita menyimpan setiap *state* dalam suatu tabel dan mengisinya seiring pergantian tahap tanpa dilakukan secara rekursif.

		j → / i ↓	0	1	2	3	4	5	6	7	8	9	10	11
		0	0	0	0	0	0	0	0	0	0	0	0	0
$\omega_1=1$	$V_1=1$	1	0	1	1	1	1	1	1	1	1	1	1	1
$\omega_2=2$	$V_2=6$	2	0	1	6	7	7	7	7	7	7	7	7	7
$\omega_3=5$	$V_3=18$	3	0	1	6	7	7	18	19	24	25	25	25	25
$\omega_4=6$	$V_4=22$	4	0	1	6	7	7	18	22	24	28	29	29	40
$\omega_5=7$	$V_5=28$	5	0	1	6	7	7	18	22	28	29	34	35	40

Gambar 2.3 Contoh *Tabulation Bottom-Up* DP

Sumber: Google Images

B. Mekanisme Pengelolaan Sumber Daya Perusahaan

Pada perusahaan berbasis proyek, sumber daya yang sangat diprihatinkan saat mengambil suatu proyek adalah jumlah ketersediaan karyawan yang dimiliki. Tentu suatu proyek yang besar membutuhkan karyawan yang lebih banyak dari proyek yang lebih kecil. Oleh karena itu, perlu dipertimbangkan juga perbandingan keuntungan yang didapatkan dan sumber daya yang digunakan.

Persoalan ini mirip sekali dengan *knapsack problem* sehingga permasalahan dapat dipecah menjadi dua komponen, yaitu jumlah anggota yang tersedia dalam perusahaan tersebut dan keuntungan yang didapatkan dari suatu proyek.

Oleh karena itu, terdapat beberapa langkah untuk persiapan pengelolaan sumber daya perusahaan.

1. Evaluasi ketersediaan sumber daya

   Pertimbangkan jumlah karyawan yang diperlukan dalam suatu perusahaan berdasarkan rata-rata jumlah proyek yang ditawarkan. Tentu semakin banyak karyawan memerlukan semakin banyak modal untuk menjalankan bisnis perusahaan.

2. Penilaian keuntungan proyek

   Setiap proyek juga perlu dievaluasi apakah keuntungan yang didapatkan sesuai dengan kompleksitas dari proyek tersebut. Jika keuntungan yang didapatkan tidak sesuai, perlu dilakukan negosiasi atau penolakkan karena keuntungan proyek juga akan digunakan untuk membayar sumber daya yang diperlukan.

3. Optimisasi penggunaan sumber daya

   Setelah ketersediaan sumber daya dan keuntungan proyek sudah dievaluasi, maka dilakukan teknik pengoptimisasian dengan algoritma yang akan dibahas pada bab berikutnya.

III. Analisis Permasalahan

A. Penggunaan Dynamic Programming dalam Optimisasi Pengelolaan Sumber Daya Perusahaan

Untuk metode program dinamis yang digunakan untuk permasalahan ini adalah *bottom-up*. Salah satu pertimbangan terbesar dalam penggunaan *bottom-up* dibandingkan *top-down* adalah untuk jumlah proyek yang semakin banyak, jumlah submasalah yang harus diselesaikan juga semakin besar sehingga algoritma rekursif akan semakin tidak memungkinkan untuk digunakan. Oleh karena itu, digunakan algoritma *tabulation* untuk kasus optimisasi pengelolaan sumber daya perusahaan.

Untuk mengilustrasikan penggunaan program dinamis pada persoalan ini, mari kita gunakan sebuah kasus. Misalkan, pada suatu perusahaan terdapat sepuluh karyawan dan ditawarkan lima buah proyek yang memiliki spesifikasi sebagai berikut.

Tabel 3.1 Tawaran Proyek

No	Nama Proyek	Kebutuhan Karyawan	Penghasilan Proyek
1	Proyek A	3	50.000.000
2	Proyek B	5	130.000.000
3	Proyek C	2	20.000.000
4	Proyek D	4	100.000.000
5	Proyek E	6	150.000.000

Karena tidak menggunakan rekursif, maka tidak perlu dibuat basis rekurens dan fungsi rekursif untuk solusi persoalan ini. Hal pertama yang dilakukan pada algoritma optimisasi ini adalah membuat tabel dengan ukuran (banyak proyek + 1) x (jumlah karyawan + 1). Lalu, tabel ini akan diisi dengan nilai 0 sebagai default value untuk setiap cell-nya. Berikut adalah ilustrasi tabel pada kondisi awal.

Tabel 3.2 DP Kondisi Awal

p\k	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0

dengan p adalah proyek ke-*i* dan *k* adalah jumlah karyawan = *j*.

Lalu, dilakukan iterasi dari proyek ke-1 hingga ke-*n* (jumlah proyek). Kita membuat kolom dan baris ke-0 sebagai basis ketika jumlah proyek yang diambil sama dengan nol atau jumlah karyawan dalam perusahaan sama dengan nol. Dalam iterasi setiap proyek, kita akan mengisi setiap kotak ke-1 hingga ke-*m* (jumlah karyawan). Periksalah apakah nilai dari iterasi *j* (kolom) pada saat ini bernilai lebih kecil dari kebutuhan karyawan pada proyek tersebut. Jika lebih kecil, maka isi nilai kotak tersebut dengan nilai kotak tepat di atasnya (proyek sebelumnya pada kolom yang sama). Jika jumlah karyawan cukup untuk kebutuhan proyek, kita dapat mempertimbangkan apakah proyek dipilih atau tidak.

Untuk menentukan apakah suatu proyek dipilih atau tidak,

1. Hitung total keuntungan sementara jika proyek tersebut dipilih yaitu menjumlahkan keuntungan proyek tersebut dengan keuntungan yang diperoleh dari sisa kapasitas (dp[i-1][j-karyawan] + profit).
2. Bandingkan keuntungan antara perhitungan total keuntungan sementara dengan total keuntungan yang diperoleh tanpa memasukkan proyek saat ini pada kolom yang sama (dp[i-1][j]).
3. Jika total keuntungan sementara lebih besar, isilah nilai dari kotak tersebut dengan total keuntungan sementara.
4. Jika tidak lebih besar, isilah nilai dari kotak tersebut dengan total keuntungan yang diperoleh tanpa memasukkan proyek saat ini (dp[i-1][j]).

Akan dijelaskan lebih detail contoh proses pengisian tabel untuk kasus ini. Untuk baris pertama yaitu 0, tidak perlu dilakukan perhitungan sebab tentunya dengan tidak mengambil proyek apapun, tidak akan didapatkan keuntungan. Untuk baris kedua yaitu 1, karena sebelumnya belum ada pengambilan proyek, maka untuk setiap sel mulai dari kolom ke-3 (karena kebutuhan karyawan proyek A adalah 3) akan diisi keuntungan sebesar 50 juta. Berikut adalah tabel dp hasil iterasi untuk baris kedua.

Tabel 3.3 DP Kondisi Hasil Iterasi Baris Kedua

p\k	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	50	50	50	50	50	50	50	50
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0

5	0	0	0	0	0	0	0	0	0	0	0

Untuk baris ketiga yaitu 2, dapat mulai membandingkan total keuntungan sementara yang diperoleh (dari menjumlahkan keuntungan proyek yang akan diambil dengan keuntungan pada sel proyek sebelumnya dengan kolom saat itu dikurang dengan jumlah karyawan yang diperlukan oleh proyek saat ini) dengan keuntungan yang diperoleh tanpa memasukkan proyek saat ini pada kolom yang sama. Misalkan untuk kolom ke-7, total keuntungan sementara sebesar 180 juta (dari 50 + 130) dibandingkan dengan 50 juta. Karena 180 juta lebih besar daripada 50 juta, diambil hasil yang lebih besar yaitu 180 juta. Berikut adalah hasil tabel *dp* untuk iterasi baris ketiga.

Tabel 3.4 DP Kondisi Hasil Iterasi Baris Ketiga

p\k	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	50	50	50	50	50	50	50	50
2	0	0	0	50	50	130	130	130	180	180	180
3	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0

Dilakukan hal yang sama juga untuk baris-baris berikutnya sesuai langkah yang telah dijelaskan dan contoh yang dipaparkan. Berikut adalah hasil tabel *dp* untuk iterasi baris keempat dan kelima.

Tabel 3.5 DP Kondisi Hasil Iterasi Baris Keempat

p\k	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	50	50	50	50	50	50	50	50
2	0	0	0	50	50	130	130	130	180	180	180
3	0	0	20	50	50	130	130	150	180	180	200
4	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0

Tabel 3.6 DP Kondisi Hasil Iterasi Baris Kelima

p\k	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	50	50	50	50	50	50	50	50
2	0	0	0	50	50	130	130	130	180	180	180
3	0	0	20	50	50	130	130	150	180	180	200
4	0	0	20	50	100	130	130	150	180	230	230
5	0	0	0	0	0	0	0	0	0	0	0

Setelah dilakukan iterasi untuk baris-baris sebelum baris akhir, akan dilakukan juga pengisian untuk baris akhir. Namun, pengisian pada baris akhir akan menentukan nilai total keuntungan yang paling optimal, yaitu nilai yang terletak pada sel terakhir. Berikut adalah ilustrasi tabel yang sudah selesai diiterasi untuk seluruh kotaknya.

Tabel 3.7 DP Kondisi Akhir

p\k	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	50	50	50	50	50	50	50	50
2	0	0	0	50	50	130	130	130	180	180	180
3	0	0	20	50	50	130	130	150	180	180	200
4	0	0	20	50	100	130	130	150	180	230	230
5	0	0	20	50	100	130	150	150	180	230	250

dengan isi dari setiap kolom dalam satuan juta (1.000.000).

Dari hasil tabel ini, keuntungan maksimum dapat diambil pada sel terakhir yaitu dp[n][m] dengan *n* adalah jumlah proyek dan *m* adalah jumlah karyawan. Pada tabel, nilai dari d[n][m] adalah 250 juta sehingga dapat diambil kesimpulan bahwa keuntungan maksimum pada kasus ini sebesar 250 juta.

B. Analisis Perbandingan Algoritma Brute Force dengan Dynamic Programming

Jika digunakan algoritma brute force, terdapat beberapa langkah yang dapat dilakukan, yaitu

1. Membuat bentuk himpunan solusi berupa sebuah tuple ($X_1, X_2, .., X_n$) dengan setiap variable-nya memiliki domain yaitu {0,1}. 0 menandakan bahwa proyek tersebut tidak diambil dan 1 menandakan bahwa proyek tersebut diambil.

2. Buatlah seluruh kemungkinan kombinasi himpunan yang bisa dibentuk. Lakukan cara ini dengan permutasi sehingga terdapat *n*! kemungkinan yang bisa dibentuk dari kombinasi himpunan solusi.

3. Untuk setiap himpunan yang terbentuk, periksalah apakah himpunan tersebut melanggar *constraint* atau tidak dengan membandingkan total jumlah karyawan yang dibutuhkan dengan jumlah karyawan dimiliki.

4. Jika melebihi, abaikan himpunan tersebut.

5. Jika tidak melebihi, hitunglah total keuntungan yang didapatkan. Bandingkan dengan keuntungan maksimum pada saat ini. Jika melebihi dari nilai maksimum, simpan nilai tersebut menjadi nilai maksimum terbaru.

6. Lakukan langkah 3-5 untuk setiap bentuk himpunan, kemudian kembalikan nilai maksimum tersebut sebagai keuntungan maksimal yang paling optimal.

Dengan algoritma *brute force*, dapat disimpulkan bahwa kompleksitasnya adalah O(n x n!) karena terdapat n! kombinasi himpunan solusi dan untuk setiap himpunan diiterasi untuk diperiksa nilai keuntungannya serta melanggar *constraint* atau tidak.

Namun, dengan program dinamis, didapatkan bahwa kompleksitas hanya O(n x m) dengan *n* adalah jumlah proyek dan *m* adalah jumlah karyawan. Hal ini dapat dibuktikan dengan pengisian tabel yang berukuran *n x m* dan setiap pengisian dapat memanfaatkan perhitungan pada sel-sel sebelumnya sehingga tidak perlu dilakukan perhitungan yang sama berkali-kali. Oleh karena itu, optimisasi ini jauh lebih efisien dibandingkan dengan *brute force*.

C. Analisis Perbandingan Algoritma Greedy dengan Dynamic Programming

Jika digunakan algoritma *greedy*, terdapat tiga pendekatan yang dapat digunakan untuk mencari solusi. Pertama, pendekatan dengan pengambilan proyek berdasarkan keuntungan. Untuk pendekatan ini, terdapat beberapa langkah yang dapat dilakukan, yaitu

1. Urutkan proyek berdasarkan keuntungan secara mengecil.
2. Iterasi proyek berdasarkan hasil proses pengurutan dimulai dari indeks ke-0.
3. Bandingkan jumlah karyawan yang dibutuhkan dengan karyawan yang dimiliki tersisa pada saat ini.
4. Jika cukup, ambil proyek tersebut menjadi proyek pilihan. Tambahkan keuntungan proyek tersebut ke dalam total keuntungan sementara dan kurangkan juga jumlah karyawan yang tersisa.
5. Lakukan proses ini hingga seluruh proyek sudah diperiksa.

Untuk pendekatan kedua adalah dengan pengambilan proyek berdasarkan jumlah karyawan yang dibutuhkan. Untuk pendekatan ini, perbedaan utama dengan pendakatan yang pertama adalah proses pengurutan dilakukan berdasarkan jumlah karyawan yang dibutuhkan oleh proyek-proyek secara membesar. Pengurutan dilakukan secara membesar dengan tujuan untuk mengambil proyek yang membutuhkan jumlah karyawan sedikit agar proyek yang dapat diambil bisa lebih banyak.

Untuk pendekatan ketiga adalah dengan pengambilan proyek berdasarkan nilai dari keuntungan / jumlah karyawan yang dibutuhkan. Algoritma *greedy* yang diterapkan juga sama seperti sebelumnya hanya perbedaannya pada proses pengurutan dilakukan berdasarkan nilai keuntungan / jumlah karyawan yang dibutuhkan secara mengecil. Digunakan rumus tersebut karena semakin besar nilai keuntungan / jumlah karyawan, semakin besar keuntungan yang diperoleh dengan jumlah karyawan yang lebih sedikit.

Dengan algoritma *greedy*, dapat disimpulkan bahwa kompleksitasnya bergantung pada proses pengurutan yang digunakan. Jika dilakukan *merge sort*, maka kompleksitasnya adalah O(n log n) karena kompleksitas yang dibutuhkan oleh *merge sort* adalah O(n log n) dan algoritma *greedy* utamanya memiliki kompleksitas O(n) sehingga kompleksitas dari keseluruhan adalah T(n) = *n log n* + *n* yang menandakan bahwa Big-Onya yaitu O(n log n).

Terlihat bahwa kompleksitas dari algoritma *greedy* sangat minim bahkan bisa saja lebih cepat dibandingkan algoritma program dinamis. Namun, salah satu pertimbangan mengapa tidak digunakannya algoritma *greedy* adalah solusi yang dihasilkan tidak selalu optimum global melainkan optimum local. Tentu perusahaan ingin memaksimal keuntungan yang bisa didapatkan sehingga algoritma ini tidak bisa diterapkan untuk kasus bisnis yang nyata. Oleh karena itu, setelah membandingkan algoritma program dinamis dengan algoritma lainnya, program dinamis dapat memberikan hasil yang optimal dan waktu eksekusi yang cepat.

IV. Aplikasi Website Pengelolaan Sumber Daya Perusahaan Berbasis Proyek

Implementasi pengelolaan sumber daya perusahaan berbasis proyek dapat berguna untuk perusahaan yang seringkali mengelola banyak proyek. Oleh karena itu, dapat dibuat sebuah aplikasi yang meminta input dari pengguna berupa jumlah karyawan yang dimiliki, serta detail dari proyek-proyek yang dikelola. Aplikasi ini dibangun dengan bahasa utamanya merupakan *typescript* sehingga pembuatan aplikasi menggunakan paradigma berorientasi objek. Berikut adalah implementasi algoritma dan aplikasi.

A. Algoritma Program Dinamis

Terdapat tiga kelas yang dapat diidentifikasikan untuk implementasi algoritmanya. Pertama, terdapat kelas *Project* untuk merepresentasikan suatu proyek yang mungkin dikerjakan oleh perusahaan tersebut. Detail dari suatu proyek adalah nama, kebutuhan karyawan, dan keuntungan yang diperoleh. Berikut adalah implementasi kelas *Project*.

```
interface Project {
    projectName: string;
    member: number;
    profit: number;
}
```

Kedua, terdapat kelas *Result* yang merepresentasikan hasil dari optimisasi pengelolaan sumber daya perusahaan. Kelas ini memiliki atribut berupa total keuntungan maksimum yang optimal dan proyek-proyek yang terpilih. Berikut adalah implementasi kelas *Result*.

```
interface Result {
    maxProfit: number;
    selectedProjects: Project[];
}
```

Ketiga, terdapat kelas *DP* yang merepresentasikan algoritma utama program dinamis yang digunakan. Berikut adalah implementasi optimisasi pengelolaan sumber daya perusahaan berbasis proyek dalam bahasa *typescript*.

```
class DP {
 optimize(projects: Project[], maxMembers: number): Result
{
  const n = projects.length;

  // Create a table to store the computed results
  const dp: number[][] = new Array(n + 1);
  for (let i = 0; i <= n; i++) {
    dp[i] = new Array(maxMembers + 1).fill(0);
  }

  // Create a table to store the selected projects
  const selected: boolean[][] = new Array(n + 1);
  for (let i = 0; i <= n; i++) {
    selected[i] = new Array(maxMembers + 1).fill(false);
  }

  // Build up the table iteratively
  for (let i = 1; i <= n; i++) {
    const { member, profit } = projects[i - 1];
    for (let j = 1; j <= maxMembers; j++) {
      if (member > j) {
     // If the current project's member exceeds the current
capacity, skip it
        dp[i][j] = dp[i - 1][j];
      } else {
     // Consider the maximum profit by either including or
excluding the current project
        const includeProfit = dp[i - 1][j - member] + profit;
        if (includeProfit > dp[i - 1][j]) {
          dp[i][j] = includeProfit;
          selected[i][j] = true; // Mark the project as selected
        } else {
          dp[i][j] = dp[i - 1][j];
        }
      }
    }
  }

  // Retrieve the selected projects
  const selectedProjects: Project[] = [];
  let i = n;
  let j = maxMembers;
  while (i > 0 && j > 0) {
    if (selected[i][j]) {
      selectedProjects.push(projects[i - 1]);
      j -= projects[i - 1].member;
    }
    i--;
  }

  console.log(dp);

  // The last cell of the table will contain the maximum profit
  return {
    maxProfit: dp[n][maxMembers],
    selectedProjects: selectedProjects.reverse(),
  };
 }
}
```

Pada implementasi algoritma di atas, terdapat dua tabel yang digunakan untuk memilih proyek-proyek. Tabel pertama yaitu *dp*, digunakan untuk menyimpan nilai-nilai perhitungan program dinamis. Tabel kedua yaitu *selected*, digunakan untuk menyimpan informasi proyek apa saja yang dipilih untuk kondisi pada saat itu sehingga tabel ini merupakan array dua dimensi yang bertipe boolean. Cara kerja algoritmanya sesuai dengan penjelasan pada bab analisis permasalahan, hanya saja ada modifikasi sedikit mengenai penyimpanan informasi proyek-proyek yang dipilih.

Tabel *dp* diinisiasi dengan diisi *default value* yaitu 0 untuk setiap selnya. Tabel *selected* juga diinisasi dengan diisi *default value* yaitu *false* untuk setiap selnya. Kemudian, seiring pengisian tabel *dp* dalam iterasi *i* dan *j*, ketika suatu proyek dipertimbangkan untuk dipilih, kita juga akan meng-set *value* dari selected[i][j] tersebut dengan *value true* yang menandakan proyek tersebut dipilih.

Setelah setiap sel berhasil dikalkulasi, diperlukan untuk mengumpulkan seluruh proyek yang telah dipilih dalam solusi. Untuk mengumpulkan proyek-proyek tersebut, akan diiterasi tabel selected dimulai dari sel terakhir yaitu dp[jumlah proyek][jumlah karyawan]. Akan diperiksa dengan mengiterasi setiap proyek secara mundur, apakah proyek tersebut diambil atau tidak. Jika diambil, akan disimpan dalam suatu array sementara. Jika tidak diambil, akan dilewati ke proyek berikutnya. Ketika solusi sudah terkumpul semua, makan dikembalikan kelas *Result* yang memiliki nilai maksimum keuntungan dan kumpulan proyek yang terpilih berdasarkan algoritma program dinamis.

B. Aplikasi Website Pengelolaan Sumber Daya Perusahaan Berbasis Proyek

Untuk pembuatan aplikasi website ini, digunakan beberapa *tools* seperti *framework* untuk *typescript* yaitu Next.js dan *framework* untuk CSS yaitu TailwindCSS. Aplikasi ini berhasil dibangun dan di-deploy ke *server deployment* yaitu Vercel. Untuk *link* dari aplikasi terdapat pada lampiran makalah ini.

Dalam merancang aplikasi website ini, pertama perlu didefinisikan *state-state* apa saja yang dimiliki pada aplikasi. Terdapat beberapa *state* yang terdefinisi untuk aplikasi ini yaitu kumpulan proyek yang ditawarkan, jumlah karyawan yang dimiliki, kumpulan proyek yang dipilih, input keuntungan, input jumlah karyawan yang diperlukan, input nama proyek, serta keadaan keberjalanan algoritma. Berikut adalah implementasi state-state aplikasi.

```
const [projects, setProjects] = useState<Project[]>([]);
const [members, setMembers] = useState(0);
```

```
const [chosenProjects, setChosenProjects] =
useState<Project[]>([]);
const [profit, setProfit] = useState(0);
const [member, setMember] = useState(0);
const [projectName, setProjectName] = useState("");
const [isOptimized, setIsOptimized] = useState(false);
```

Terdapat beberapa fitur yang diimplementasikan pada aplikasi *website* ini. Pertama, fitur untuk mengisi jumlah karyawan yang dimiliki oleh perusahaan. Jumlah ini akan disimpan dalam *state* "members". Berikut adalah implementasi fitur tersebut.

```
<div className="flex flex-col gap-2 items-center mt-4">
  <label>Available Members</label>
  <input
    type="number"
    value={members}
    className="text-center p-2 text-black"
     onChange={(e) =>
setMembers(Number(e.target.value))}
        />
</div>
```

Kedua, fitur untuk menambahkan proyek baru. Untuk menambahkan proyek baru, pengguna dapat menginput nama proyek, jumlah karyawan yang dibutuhkan, serta total keuntungan yang diperoleh dari proyek tersebut. Dibuatkan fungsi untuk mengolah input proyek dari pengguna apakah *valid* atau tidak. Jika valid, maka ditambahkan kepada state "projects". Berikut adalah implementasi fitur tersebut.

```
  <form className="mt-12" onSubmit={(e) =>
submitProject(e)}>
        <div className="flex flex-col items-center gap-4">
          <h1 className="text-3xl font-semibold text-
center">
            New Project
          </h1>
          <div className="flex flex-col gap-2">
            <label>Project Name</label>
            <input
              type="text"
              value={projectName}
              className="p-2 text-black"
              onChange={(e) =>
setProjectName(e.target.value)}
            />
            <label>Profit</label>
            <input
              type="number"
              value={profit}
              className="p-2 text-black"
              onChange={(e) =>
setProfit(Number(e.target.value))}
            />
            <label>Needed Members</label>
            <input
              type="number"
              value={member}
              className="p-2 text-black"
              onChange={(e) =>
setMember(Number(e.target.value))}
            />
          </div>
          <button type="submit" className="border-2 px-4
py-2">
            Submit
          </button>
        </div>
      </form>
```

```
const submitProject = (e:
 FormEvent<HTMLFormElement>) => {
  e.preventDefault();
  // Validate input
  if (projectName === "" || profit === 0 || member === 0) {
    return;
  }

  // Add new project
  let temp = [...projects];
  temp.push({ projectName: projectName, member:
 member, profit: profit });
  setProjects(temp);

  // Reset input
  setProjectName("");
  setMember(0);
  setProfit(0);

  // Optimize
  if (isOptimized) {
    // Validate projects
    if (temp.length === 0 || members === 0) {
      return;
    }

    // Optimize with DP
    let optimize = new DP();
    let result: Result = optimize.optimize(temp, members);
    setChosenProjects(result.selectedProjects);
    setIsOptimized(true);
  }
};
```

Terakhir, yaitu fitur untuk mencari pilihan proyek yang memberikan keuntungan maksimum dan tombol reset untuk mereset seluruh keadaan aplikasi menjadi semula. Untuk melakukan perhitungan keuntungan maksimum, akan di-*construct* kelas *DP* untuk memanggil metode yang mengaplikasikan algoritma program dinamis yang telah didefinisikan sebelumnya. Untuk melakukan reset, akan dilakukan pengubahan seluruh nilai *state* menjadi *default value*. Berikut adalah implementasinya.

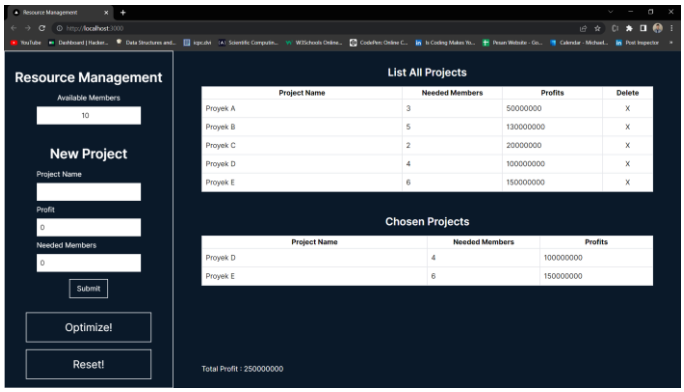
```
 const optimizeResource = (e:
FormEvent<HTMLButtonElement>) => {
    e.preventDefault();
```

```
    // Validate projects
    if (projects.length === 0 || members === 0) {
      return;
    }

    // Optimize with DP
    let optimize = new DP();
    let result: Result = optimize.optimize(projects,
members);
    setChosenProjects(result.selectedProjects);
    setIsOptimized(true);
  };
```

```
const reset = (e: FormEvent<HTMLButtonElement>) => {
    e.preventDefault();
    // Reset
    setProjects([]);
    setChosenProjects([]);
    setIsOptimized(false);
  };
```

Maka, telah berhasil dibangun aplikasi website untuk persoalan makalah ini. Aplikasi ini dapat diakses oleh pengguna umum secara online melalui link yang dilampirkan pada bab lampiran. Berikut adalah contoh tampilan aplikasi website secara utuh.



Gambar 4.1 Tampilan Aplikasi Website

Sumber: Pribadi

V. UJI KASUS

Untuk memastikan aplikasi dapat berjalan dengan baik dan sesuai dengan harapan, akan diuji dengan beberapa uji kasus. Berikut adalah beberapa uji kasus dan solusi yang dihasilkan oleh aplikasi.

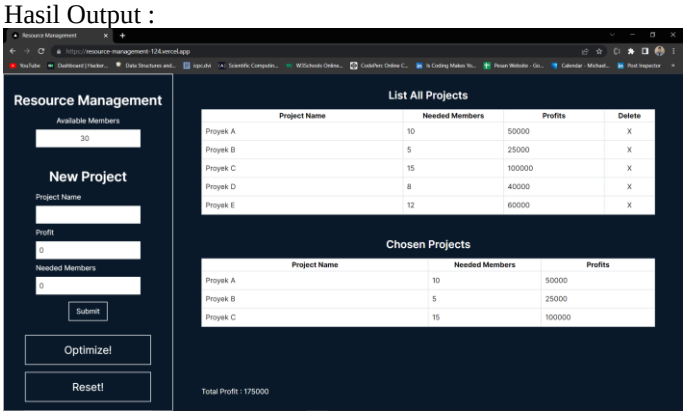
A. Uji Kasus Pertama

Jumlah Karyawan : 30

Tabel 5.1 Daftar Proyek Uji Kasus Pertama

Nama Proyek	Jumlah Karyawan	Keuntungan
Proyek A	10	$50,000
Proyek B	5	$25,000
Proyek C	15	$100,000
Proyek D	8	$40,000
Proyek E	12	$60,000

Hasil Output :



Gambar 5.1 Hasil Output Uji Kasus Pertama

Sumber: Pribadi

Tabel 5.2 Hasil Optimal Uji Kasus Pertama

Nama Proyek	Jumlah Karyawan	Keuntungan
Proyek A	10	$50,000
Proyek B	5	$25,000
Proyek C	15	$100,000
Keuntungan		$175,000

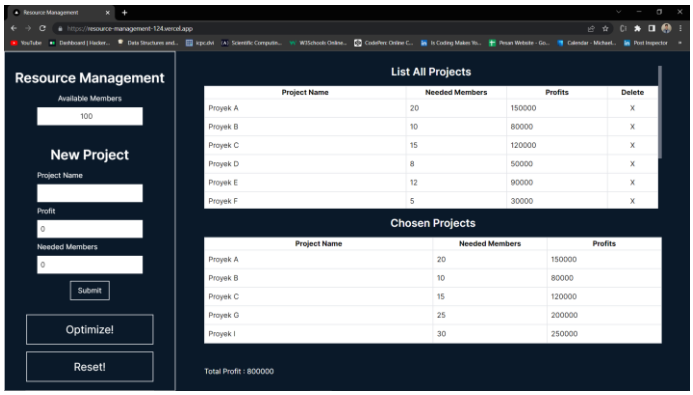
B. Uji Kasus Kedua

Jumlah Karyawan : 100

Tabel 5.3 Daftar Proyek Uji Kasus Kedua

Nama Proyek	Jumlah Karyawan	Keuntungan
Proyek A	20	$150,000
Proyek B	10	$80,000
Proyek C	15	$120,000
Proyek D	8	$50,000
Proyek E	12	$90,000
Proyek F	5	$30,000
Proyek G	25	$200,000
Proyek H	18	$140,000
Proyek I	30	$250,000
Proyek J	22	$180,000

Hasil Output :



Gambar 5.2 Hasil Output Uji Kasus Kedua

Sumber: Pribadi

Tabel 5.4 Hasil Optimal Uji Kasus Kedua

Nama Proyek	Jumlah Karyawan	Keuntungan
Proyek A	20	$150,000
Proyek B	10	$80,000
Proyek C	15	$120,000
Proyek G	25	$200,000
Proyek I	30	$250,000
Keuntungan		$800,000

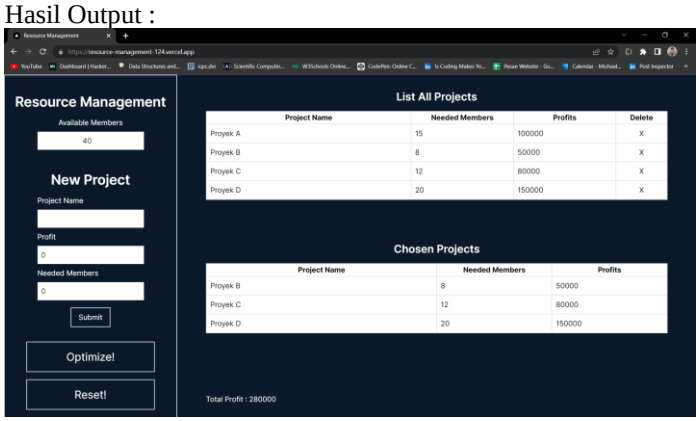
C. Uji Kasus Ketiga

Jumlah Karyawan : 40

Tabel 5.5 Daftar Proyek Uji Kasus Ketiga

Nama Proyek	Jumlah Karyawan	Keuntungan
Proyek A	15	$100,000
Proyek B	8	$50,000
Proyek C	12	$80,000
Proyek D	20	$150,000

Hasil Output :



Gambar 5.3 Hasil Output Uji Kasus Ketiga

Sumber: Pribadi

Tabel 5.6 Hasil Optimal Uji Kasus Ketiga

Nama Proyek	Jumlah Karyawan	Keuntungan
Proyek B	8	$50,000
Proyek C	12	$80,000
Proyek D	20	$150,000
Keuntungan		$280,000

VI. Kesimpulan Dan Saran

Algoritma program dinamis dapat digunakan untuk pencarian solusi yang optimum global seperti contohnya pada kasus ini, program dinamis digunakan untuk mencari keuntungan maksimum yang paling optimal. Walaupun terdapat beberapa teknik lainnya untuk mencari solusi seperti *brute force* dan *greedy*, namun keduanya tidak efisien untuk kebutuhan bisnis karena *brute force* akan memakan banyak resource komputasi dan waktu yang lebih lama sedangkan *greedy* tidak selalu menghasilkan solusi yang optimum global.

Namun, masih terdapat kelemahan pada progam ini seperti tidak diimplementasikan basis data untuk menyimpan hasil optimisasi yang dibutuhkan pengguna sehingga aplikasi hanya menampilkan solusi secara sementara saja. Alangkah baiknya jika program dapat memiliki sistem login dan basis data agar aplikasi dapat digunakan lebih baik oleh pengguna.

VII. Ucapan

Saya panjatkan puji dan syukur kepada Tuhan Yang Maha Esa karena atas segala berkat dan rahmat-Nya, makalah ini dapat diselesaikan. Saya selaku penulis mengucapkan terima kasih kepada Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc. sebagai dosen pengampu mata kuliah IF2211 Strategi Algoritma yang telah mengajari penulis. Mohon maaf bila ada kesalahan dalam pembuatan makalah ini. Semoga makalah ini dapat berguna bagi pembaca.

VIII. Lampiran

Github Repository Project Link

https://github.com/maikeljh/makalah-stima-resource-management

Website Link

https://resource-management-124.vercel.app/

Video Link at Youtube

https://youtu.be/Tqg2QCbl8xk

References

[1] Munir, Rinaldi. 2020. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf, diakses pada 13 Mei 2023.

[2] Munir, Rinaldi. 2020. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian2.pdf, diakses pada 13 Mei 2023.

[3] GeeksForGeeks. 2023. https://www.geeksforgeeks.org/dynamic-programming/, diakses pada 13 Mei 2023.

[4] Section, 2022. https://www.section.io/engineering-education/dynamic-programming-in-javascript-using-tabulation/, diakses pada 13 Mei 2023.

[5] UMA, 2021. http://akuntansi.uma.ac.id/2021/08/27/organisasi-berbasis-proyek-dan-pentingnya/, diakses pada 13 Mei 2023.

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, 22 Mei 2023

Michael Jonathan Halim
13521124