Program Dinamis dan Pencocokan String untuk Penyelesaian Soal Pemrograman Kompetitif

Christian Anthony Setyawan - 13514085
ITB
Bandung - Indonesia
13514085@std.stei.itb.ac.id

***Abstrak*—Program dinamis dan pencocokan string dapat menyelesaikan berbagai masalah. Salah satu yang dapat dipecahkan menggunakan keduanya adalah masalah Party at Hali-Bula, sebuah soal dari lomba ACM ICPC regional Tehran, Asia tahun 2006. Soal ini menginginkan jumlah maksimal orang yang dapat diundang tanpa boss dari orang tersebut juga ikut. Selain itu juga tentukan apakah terdapat konfiigurasi pengundangan yang berbeda untuk jumlah maksimal tersebut.**

Kata Kunci—program dinamis; pencocokan string; acm icpc; pemrograman kompetitif; hali-bula

I. PENDAHULUAN

Program dinamis (dynamic programming) yang ditemukan oleh Richard Bellman pada tahun 1953 merupakan suatu metode penyelesaian masalah di mana solusi persoalan dapat dipandang sebagai serangkaian keputusan yang saling berkaitan. Program dinamis merupakan salah satu metode yang mangkus yang biasanya digunakan untuk menyederhanakan persoalanpersoalan rekursif.

Seperti halnya algoritma greedy, program dinamis juga merupakan suatu ancangan untuk menyelesaikan masalah optimasi. Hanya saja, pada metode greedy hanya satu rangkaian keputusan yang pernah dihasilkan, sedangkan dengan program dinamis lebih dari satu rangkaian keputusan.

Program dinamis fokus pada bagian permasalahan yang tumpang-tindih (overlapping subproblems). Rangkaian keputusan dibuat dengan prinsip optimalitas (optimal substructure), di mana solusi optimal dari bagian solusi permasalahan bisa digunakan untuk menemukan solusi optimal untuk masalah secara keseluruhan.

Penerapan program dinamis ini sangat luas. Di antaranya, yang sederhana, adalah untuk menghitung angka Fibonacci dan koefisien binomial. Juga terdapat sejumlah algoritma lain yang didasarkan pada program dinamis, seperti algoritma Cocke-Younger-Kasami (CYK) untuk menentukan apakah suatu string dapat diterima suatu context-free grammar, algoritma Viterbi untuk model hidden Markov, algoritma Earley untuk chart parser, algoritma Needleman-Wunsch yang dipakai dalam bioinformatik, algoritma Floyd's All-Pairs untuk mencari lintasan terpendek, algoritma Selinger untuk optimasi query suatu basis data, algoritma De Boor untuk evaluasi kurva B-spline, dan lain-lain.

Pemrograman kompetitif adalah sebuah lomba premrograman yang biasanya diadakan melalui Internet atau jaringan lokal, yang melibatkan peserta mencoba program sesuai dengan spesifikasi spesifik yang disediakan untuk meyelesaikan masalah yang diberikan. Biasanya masalah yang diberikan tersebut memiliki keterkaitan dengan masalah yang ada di dunia nyata. Pemrograman kompetitif diakui dan didukung oleh beberapa perusahaan perangkat lunak dan Internet multinasional, seperti Google, Facebook dan IBM. Ada beberapa organisasi yang menjadi tuan rumah untuk kompetisi pemrograman secara teratur.

Sebuah kompetisi pemrograman umumnya melibatkan tuan rumah untuk menyajikan serangkaian masalah logis atau matematis, termasuk salah satunya pemrograman dinamis, untuk para kontestan (yang dapat bervariasi dalam jumlah puluhan hingga beberapa ribu), dan kontestan diwajibkan untuk menulis program komputer yang mampu memecahkan setiap masalah. Dilihat didasarkan sebagian besar pada sejumlah masalah diselesaikan dan waktu yang dihabiskan untuk menulis solusi sukses, tetapi mungkin juga termasuk faktor-faktor lain (kualitas output yang dihasilkan, waktu eksekusi, ukuran program, dll)

ACM International Collegiate Programming Contest (disingkat ACM-ICPC atau hanya ICPC) adalah kompetisi pemrograman tahunan multi-tier kompetitif di antara universitas-universitas di dunia. Kontes ini disponsori oleh IBM. Berkantor pusat di Baylor University, dengan daerah otonom di enam benua, yang ICPC diarahkan oleh Baylor Profesor William B. Poucher, Direktur Eksekutif, dan beroperasi di bawah naungan Association for Computing Machinery (ACM).

II. DASAR TEORI

A. Program Dinamis

Program dinamis (Dynamic Programming) adalah salah satu metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah atau tahapan sedemikian

sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan.

Menurut Dimayati (1992), program dinamis adalah suatu teknik matematis yang biasanya digunakan untuk membuat suatu keputusan dari serangkaian keputusan yang saling berkaitan. Tujuan utama model ini adalah untuk mempermudah penyelesaian persoalan optimasi yang mempunyai karakteristik tertentu. Istilah program dinamis, pertama kali diperkenalkan pada era 1950-an oleh Richard Bellman seorang professor di Universitas Princeton dan juga bekerja pada RAND corporation, perlu diketahui bahwa RAND corporation pada era itu merupakan suatu perusahan yang dibentuk untuk menawarkan analisis dan riset untuk angkatan bersenjata Amerika Serikat. Saat itu, Bellman bekerja di bidang pengambilan keputusan multi tahap (multistage desicion process) dan mengerjakan beberapa metode matematis, beberapa tahun kemudian setelah Bellman berada di RAND, lahirlah istilah pemrograman dinamis. Istilah ini tidak secara langsung berhubungan dengan pemrograman, melainkan digunakan sebagai judul project yang kemudian yang diusulkan RAND coorporation pada Angkatan Udara Amerika Serikat.

Ada beberapa hal mendasar yang perlu diketahui pada penyelesaian persoalan dengan program dinamis ini, yaitu:

1. Terdapat sejumlah berhingga pilihan yang mungkin,
2. Solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya,
3. Menggunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

Penyelesaian persoalan dengan metode program dinamis ini memiliki kemiripan dengan metode greedy yang juga membentuk solusi secara bertahap. Pada metode greedy, pengambilan keputusan pada setiap tahap dilakukan dengan cara mengambil pilihan yang paling menarik (memenuhi ukuran optimasi yang digunakan), didasarkan hanya pada informasi lokal, dan pada setiap tahap itu tidak pernah diambil keputusan yang salah. Sebagai contoh, pada persoalan minimisasi waktu di dalam sistem, dimana pelanggan berikutnya yang akan dimasukkan ke dalam antrian adalah pelanggan yang membutuhkan waktu pelayanan terkecil diantara pelanggan lain yang belum dilayani. Pada persoalan tertentu, algoritma greedy bekerja dengan baik, namun pada kebanyakan persoalan yang lain tidak. Hal tersebut karena pengambilan keputusan pada setiap langkah greedy tidak pernah mempertimbangkan lebih jauh apakah pilihan tersebut merupakan pilihan yang tepat pada langkah-langkah selanjutnya.

Berbeda dengan greedy, program dinamis menggunakan Prinsip Optimalitas untuk membuat rangkaian keputusan yang optimal. Prinsip tersebut berbunyi: jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal. Prinsip optimalitas berarti bahwa jika kita bekerja dari tahap k ke tahap k+1, maka kita dapat menggunakan hasil optimal dari tahap k tanpa harus kembali ke tahap awal. Jika ongkos (cost) pada setiap tahap dihitung, maka dapat dirumuskan bahwa

Ongkos pada tahap k+1 = (ongkos yang dihasilkan pada tahap k) + (ongkos dari tahap k ke tahap k+1)

Dengan prinsip optimalitas, bisa dijamin bahwa pengambilan keputusan pada suatu tahap adalah keputusan yang benar untuk tahap-tahap selanjutnya. Jadi, perbedaan mendasar antara greedy dengan program dinamis adalah bahwa hanya ada satu rangkaian keputusan yang pernah dihasilkan pada metode greedy, sedangkan pada program dinamis ada lebih dari satu rangkaian keputusan, dan hanya rangkaian keputusan yang memenuhi prinsip optimalitas yang akan dihasilkan.

Nama program dinamis sendiri mengacu pada perhitungan dengan menggunakan tabel. Seperti halnya algoritma greedy, program dinamis digunakan untuk memecahkan masalah optimasi.First, confirm that you have the correct template for your paper size. This template has been tailored for output on the A4 paper size. If you are using US letter-sized paper, please close this file and download the file “MSW_USltr_format”.

B. Karakteristik Program Dinamis

Program dinamis diterapkan pada persoalan yang memiliki karakteristik sebagai berikut:

1. Persoalan dapat dibagi menjadi beberapa tahap (stage), dan pada setiap tahap hanya diambil satu keputusan.
2. Masing - masing tahap terdiri dari sejumlah status (state) yang berhubungan dengan tahap tersebut. Status merupakan berbagai kemungkinan masukan yang ada pada tahap tersebut. Jumlah status bisa berhingga atau tidak berhingga.
3. Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya.
4. Ongkos (cost) pada suatu tahap meningkat secara teratur dengan bertambahnya jumlah tahapan.
5. Ongkos pada suatu tahap bergantung pada ongkos tahap – tahap yang sudah berjalan dan ongkos pada tahap tersebut.
6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan yang dilakukan pada tahap sebelumnya.
7. Adanya hubungan rekursif yang mengidentifikasikan keputusan terbaik untuk setiap status pada tahap k memberikan keputusan terbaik untuk setiap status pada tahap k+1.
8. Prinsip optimalitas berlaku pada persoalan tersebut.

Pohon adalah graf tak berarah terhubung yang tidak mengandung sirkuit. Kumpulan dari pohon-pohon yang saling lepas disebut dengan hutan. Pohon memiliki sifar antara lain:

- Setiap pasang simpul dalam pohon terhubung dengan lintasan tunggal
- Pohon adalah graf yang terhubung dan jika memiliki n buah simpul maka memiliki n-1 buah sisi

- Pohon tidak mengandung sirkuit, yang artinya ada lintasan yang berawal dan berakhir di simpul yang sama
- Pohon tidak mengandung sirkuit, dan penambahan satu sisi di pohon , akan membuat adanya sirkuit dan hanya terdapat satu sirkuit saja.
- Pohon terhubung dan semua sisianya adalah jembatan.

Pohon juga memiliki struktur dalam implementasinya antara lain:

- Akar adalah bagian puncak dari suatu pohon, sebagai suatu sumber. Dapat juga dikatakan sebagai hulu.
- Anak adalah percabangan pertama dari suatu simpul.
- Parent atau orangtua adalah orangtua dari suatu simpul , atau yang memperanakkan simpul tersebut
- Saudara kandung adalah simpul yang memiliki orangtua / parent yang sama
- Daun adalah simpul yang tidak memiliki anak atau berderajat nol.

Terdapat juga istilah-istilah yang digunakan oleh pohon antara lain:

- Simpul adalah bagian dari pohon yang berisi informasi yang dihubungkan dengan cabang antara simpul yang satu ke simpul yang lain
- Lintasan langkah yang ditempuh berupa simpul maupun cabang dari suatu simpul ke simpul lain
- Derajat adalah jumlah anak yang dimiliki oleh suatu simpul
- Simpul dalam adalah simpul yang memiliki anak
- Aras adalah level/tingkat dari suatu simpul dari pohon
- Tinggi atau kedalaman adalah aras maksimum yang dimiliki oleh simpul dari suatu pohon
- Upapohon adalah sub bagian dari suatu pohon

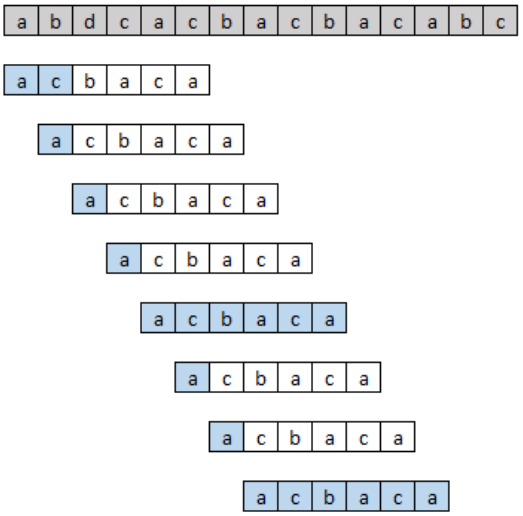
Terdapat juga istilah pohon merentang. Yaitu upagraf yang didapatkan dari graf terhubung yang membentuk pohon. Pohon merentang dapat diperoleh dengan memutus sirkuit di dalam graf. Setiap graf terhubung memiliki satu buah pohon merentang. Graf tak terhubung dengan k komponen mempunyai k buah hutan merentang yang disebut hutan merentang. Pohon berakar adalah pohon yang satu buah simpulnya diberlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah

C. Pencocokan String

Pencarian string dalam sebuah teks biasa disebut dengan pencocokan string (string matching atau pattern matching). Dalam pencocokan string, tulisan di mana pencarian dilakukan dinamakan sebagai teks dengan panjang n. Lalu, string yang dicari disebut sebagai pattern dengan panjang m, di mana pattern memenuhi syarat panjang m kurang dari panjang n (m<n). Terdapat beberapa algoritma yang dapat digunakan untuk melakukan pencocokan string yaitu algoritma Brute Force, algoritma Knuth-Morris-Pratt, dan algoritma Boyer-Moore.

Algoritma Brute Force

Brute force merupakan sebuah pendekatan dalam pemecahan masalah yang penyelesaiannya lempeng (straight forward). Pencocokan string dengan algoritma brute force dapat dilakukan dengan mencocokkan satu per satu karakter yang ada pada pattern dengan karakter yang ada di teks. Jika ditemukan ketidakococokan, geser pattern sejauh satu karakter, lalu ulangi lagi pencocokan dari karakter pertama pada pattern.



Gambar 1. Ilustrasi pencocokan string dengan algoritma brute force.

Gambar di atas merupakan ilustrasi dari pencocokan string menggunakan algoritma Boyer-Moore. Karakter dengan warna biru merupakan karakter pada pattern yang dibandingkan dengan karakter pada teks. Kasus terburuk pada pencocokan string menggunakan algoritma brute force adalah ketika karakter awal pada pattern cocok dengan teks, namun karakter terakhirnya tidak cocok dan kecocokan baru ditemukan pada karakter terakhir dari teks. Kompleksitas algoritmanya pada kasus terburuk adalah O(mn).

Contoh:

teks : aaaaaaaaaaaaaaah

pattern : aah

Kasus terbaik dapat dicapai pada kondisi karakter pertama pada pattern tidak pernah mengalami kecocokan dengan teks sebelum pattern ditemukan. Komplekstitas algoritma pada kasus ini adalah O(n).

Contoh:

teks: Pada hari Minggu saya membeli kacang

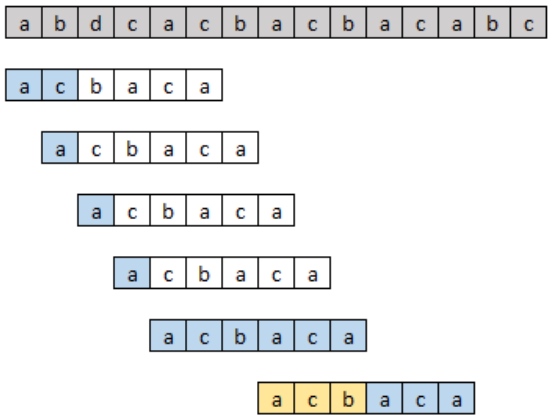
pattern: kacang

Kasus yang paling sering terjadi (kasus rata-rata) yaitu ketikaada beberapa karakter yang cocok pada pattern da nada beberapa yang cocok. Kompleksitas algoritmanya adalah

O(m+n). Algoritma brute force akan optimal ketika jumlah alfabet besar dan akan kurang optimal ketika jumlah alfabet kecil.

Algoritma Knuth-Morris-Pratt

Pencocokan string pada algoritma Knuth-Morris-Pratt (KMP) dilakukan dari kiri. Algoritma KMP merupakan perbaikan dari algoritma brute force. Perbaikan dalam algoritma ini terdapat pada penggeseran pattern jika terjadi ketidakcocokan. Pergeseran pattern ditentukan oleh sebuah fungsi yang dinamakan fungsi pinggiran. Fungsi pinggiran merupakan fungsi yang mengembalika jumlah awalan (prefix) dari pattern yang juga merupakan suffix.



Gambar 2. Ilustrasi pencocokan string dengan algoritma Knuth-Morris-Pratt.

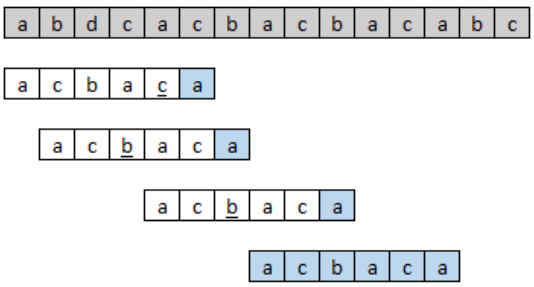
Gambar di atas merupakan ilustrasi dari pencocokan string menggunakan algoritma KMP. Karakter dengan warna biru merupakan karakter pada pattern yang dibandingkan dengan karakter pada teks. Karakter dengan warna kuning merupakan karakter yang sudah tidak perlu dibandingkan lagi karena adanya fungsi pinggiran.

Kompleksitas yang dibutuhkan untuk mencari fungsi pinggiran adalah O(m) dan kompleksitas yang dibutuhkan untuk pencocokan string adalah O(n). Sehingga, kompleksitas dari algoritma KMP ini adalah O(m+n). Kasus terbaik akan terjadi jika jumlah alfabet pada teks kecil.

Algoritma Boyer-Moore

Algoritma Boyer-Moore melakukan pencocokan string dari kanan (dari belakang). Terdapat dua karakeristik dari algoritma Boyer-Moore. Pertama yaitu teknik lookingglass yang merupakan teknik pencocokan yang dilakukan dari karakter paling belakang pada pattern. Kedua, yaitu teknik character-jump yang dilakukan apabila terjadi ketidakcocokan pada karakter. Terdapat tiga kondisi yang menentukan character jump:

1. Karakter pada teks ditemukan di sisi kiri karakter pada pattern. Jika hal ini terjadi, maka geser pattern sampai karakter yang sama yang ada pattern bersesuaian posisinya dengan karakter pada teks
2. Karakter pada teks ditemukan di sisi kanan karakter pada pattern. Apabila hal ini terjadi, maka geser pattern sejauh satu karakter.
3. Karakter pada teks tidak ditemukan pada pattern. Jika konsisi ini terjadi, geser pattern hingga melewati karakter yang tidak cocok pada teks.



Gambar 3. Ilustrasi pecocokan string dengan algoritma Boyer-Moore.

Gambar di atas merupakan ilustrasi dari pencocokan string menggunakan algoritma Boyer-Moore. Karakter dengan warna biru merupakan karakter pada pattern yang dibandingkan dengan karakter pada teks. Karakter yang digaris bawahi merupakan karakter pada patternyang sama dengan karakter pada teks saat terjadi ketidakcocokan antara pattern dengan teks.

Kasus terburuk pada algoritma Boyer-Moore dicapai ketika jumlah alfabet besar, sehingga algoritma ini baik digunakan untuk melakukan pencocokan string pada tulisan biasa dan kurang cocok untuk digunakan pada pencocokan string bilangan biner.

D. Deskripsi Masalah

”Dear Contestant, I'm going to have a party at my villa at Hali-Bula to celebrate my retirement from BCM. I wish I could invite all my co-workers, but imagine how an employee can enjoy a party when he finds his boss among the guests! So, I decide not to invite both an employee and his/her boss. The organizational hierarchy at BCM is such that nobody has more than one boss, and there is one and only one employee with no boss at all (the Big Boss)! Can I ask you to please write a program to determine the maximum number of guests so that no employee is invited when his/her boss is invited too? I've attached the list of employees and the organizational hierarchy of BCM.

Best, -Brian Bennett

P.S. I would be very grateful if your program can indicate whether the list of people is uniquely determined if I choose to invite the maximum number of guests with that condition.”

Kemudian kita diberikan n buah nama orang, dengan n kurang dari sama dengan 200 dan setiap nama merupakan satu kata serta nama tersebut kurang dari 100 huruf. Diberikan juga relasi boss/karyawan orang tersebut sehingga relasi tersebut membentuk sebuah tree dengan satu orang di paling atas (*Big Boss*) yang tidak memiliki boss siapa pun.

Output yang diminta ialah jumlah orang terbanyak yang dapat diundang ke pesta tersebut sehingga tidak ada relasi boss/karyawan yang ada didalam orang yang diundang tersebut. Selain itu tentukan juga (*Yes/No*) apakah terdapat konfigurasi pengundangan yang unik sehingga orang yang diundang semaksimal mungkin dengan mengikuti aturan diatas.

III. PENYELESAIAN MASALAH

Penyelesaian masalah ini dapat dibagi menjadi 3 buah bagian berbeda: pembuatan pohon boos/karyawan, pencarian jumlah maksimum, dan penentuan solusi unik.

A. Pembuatan Pohon Boss/Karyawan

Dari input yang diberikan, yakni nama karyawan dan relasi boss/karyawan tersebut dalam pasangan nama, maka diperlukan pencocokan string untuk membuat pohon hirarki boss/karyawan tersebut. Pohon ini diperlukan untuk program dinamis yang digunakan untuk mencari jumlah maksimum orang yang dapat diundang.

Untuk setiap pasangan nama yang diberikan, kita melakukan pencocokan string terhadap kedua nama tersebut ke koleksi nama-nama yang telah diberikan di pasangan sebelum-sebelumnya. Apabila terdapat nama yang belum pernah muncul sebelumnya, maka kita membuat node pohon baru yang memiliki nama tersebut. Apabila tidak, maka hubungkan kedua node yang memiliki nama tersebut.

Dikarenakan jumlah karakter dari setiap nama relatif sedikit dan juga string yang dicocokan adalah nama, maka penulis merasa pencocokan string dapat dilakukan dengan brute force saja. Sehingga kompleksitas untuk pembuatan pohon O(nk) dengan k adalah rata-rata panjang nama di input.

B. Pencarian Jumlah Maksimum

Setelah kita mendapatkan pohon hirarki boss/karyawan, kita dapat mencari jumlah maksimum karyawan yang dapat diundang tanpa memiliki relasi langsung di pohon hirarki tersebut. Kita bisa juga menyebut masalah ini sebagai graph coloring problem, namun dengan 2 warna, hitam dan putih, kemudian ditanyakan berapa jumlah maksimal warna putih yang bisa ada.

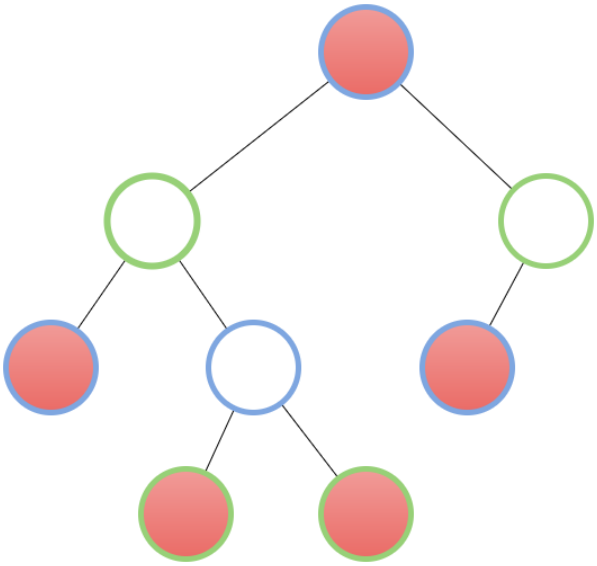
Pendekatan yang paling intuitif ialah untuk melakukan brute force dengan mencoba setiap kemungkinan karyawan yang dapat diundang. Setiap mencoba, cek apakah pemgambilan karyawan ini menyalahi aturan bahwa boss/karyawan tidak boleh diundang sekaligus. Untuk setiap pengecekan, dilakukan iterasi terhadap semua karyawan yang diambil, dan setiap karyawan dilakukan iterasi untuk setiap tetangga dari karyawan tersebut apakah tetangga tersebut dapat diambil. Meskipun terlihat ada 2 lapis iterasi, namun terlihat sebuah observasi bahwa hanya akan ada 2 kali pengecekan dalam satu edge. Oleh karena berbentuk pohon, maka pengecekan dilakukan dengan kompleksitas O(n). Brute force setiap kemungkinan pengambilan dilakukan dengan kompleksitas $O(2^n)$ sehingga kompleksitas total $O(n*2^n)$.

Namun algoritma brute force tidak cocok untuk menyelesaikan masalah ini. Kompleksitas yang bukan polynomial menjadikan algoritma ini sangat lambat ketika dihadapkan dengan input yang banyak. Tentu tidak cocok ketika konstrain dari masalah ini adalah dengan n = 200.

Pendekatan kedua untuk mengurangi kompleksitas non-polynomial yang diberikan oleh pendekatan brute force, ialah dengan melakukan pendekatan greedy. Prinsip dari greedy dari pendekatan ini ialah untuk mengambil karyawan sebanyak-banyaknya yang mungkin.

Dari observasi bahwa boss/karyawan tidak boleh ada yang diundang bersamaan, maka kita dapat mengetahui bahwa apabila semua node di level tertentu diambil, maka semua node dilevel berikut dan sebelumnya tidak akan diambil. Dengan demikian kita algoritma greedy yang kita peroleh adalah membandingkan apabila semua node di level 1 diambil, level 3 diambil, dan seterusnya dengan apabila semua node di level 2 diambil, level 4 diambil, dan seterusnya. Kompleksitas total dari algoritma greedy ini adalah O(n), dikarenakan cukup untuk iterasi disetiap node di pohon yang telah dibuat apakah ia masuk ke level genap atau ganjil.

Namun algoritma greedy meskipun sangat bagus dalam kompleksitas, tetapi tidak dapat menghasilkan jawaban yang benar untuk setiap kasus. Sebagai contoh:



Gambar 4. Ilustrasi Kesalahan Algoritma Greedy

Dari gambar diatas, algoritma greedy akan mendapatkan hasil 4, yakni menghitung jumlah lingkaran biru atau lingkaran hijau(level ganjil dan genap). Tetapi ada jumlah yang lebih banyak yakni 5, ditandai dengan lingkaran berisi merah. Oleh karena itu algoritma greedy bukanlah solusi yang baik.

Pendekatan yang terakhir ialah dengan menggunakan algoritma pemrograman dinamis. Observasi dari pendekatan brute force ialah bahwa masalah ini dapat dipecahkan secara modular, yakni ketika sebuah subgraph dengan root dari subgraph tersebut di tentukan untuk diambil atau tidak, maka ia menjadi sebuah sub-problem yang sama dengan problem sesungguhnya. Sehingga dapat dirumuskan apabila f(n, x) merupakan jumlah maksimal karyawan yang diambil di

subgraph dengan root n, dan x=1 apabila root tersebut diambil; x=0 apabila root tersebut tidak diambil, maka:

$$f(n,x) = \begin{cases} 0 & n = leaf \\ \sum_{i \in child(n)} max(f(i,0), f(i,1)+1) & x = 0 \\ \sum_{i \in child(n)} f(i,0) & x = 1 \end{cases}$$

Kompleksitas dari program dinamis ini ialah $O(n^2)$ karena terdapat 2*n state dengan fungsi diatas, dan setiap state dilakukan iterasi dengan maksimal n kali.

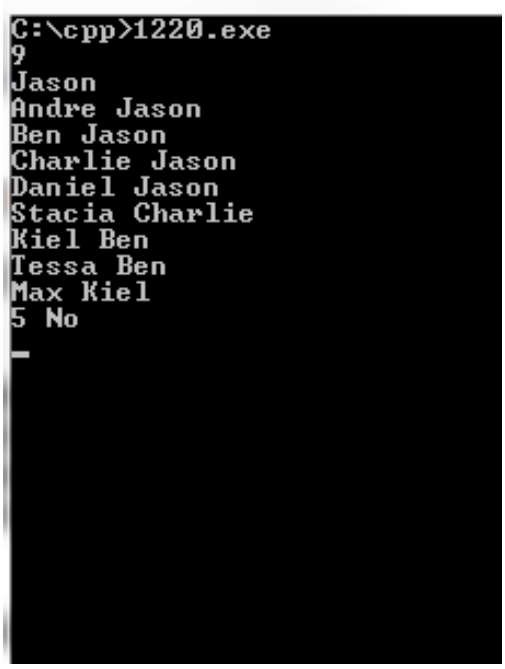
C. Penentuan Solusi Unik

Untuk menentukan apakah solusi yang diperoleh dari program dinamis tersebut unik didapat dari konfigurasi yang unik atau tidak, diperlukan sebuah obervasi sekali lagi. Solusi tidak akan unik apabila sebuah subgraph dengan root dari subgraph tersebut diambil memiliki jumlah maksimal yang sama apabila root tersebut tidak diambil.

Dengan demikian untuk menentukan solusi unik atau tidak cukup dilakukan pengecekan saja ketika mencari max yang ada di program dinamis. Apabila terdapat ada yang sama di hasil akhir, maka solusi tidak unik.

IV. KESIMPULAN

Kita bisa membuat program dinamis dan pencocokan string untuk menyelesaikan masalah tersebut. Masalah ini merupakan salah satu contoh soal dari pemrograman kompetitif. Kode asli dari penulis dapat dilihat di http://ideone.com/2DqOeX. Contoh program:



UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan yang Maha Esa, karena atas karunia-Nya makalah ini dapat diselesaikan. Penulis juga mengucapkan terima kasih kepada Bapak Rinaldi Munir dan Ibu Nur Ulfa Maulidevi selaku dosen dari mata kuliah IF2211 Strategi Algoritma yang telah membantu dan memudahkan dalam pembuatan makalah ini.

REFERENSI

[1] https://uva.onlinejudge.org/external/12/1220.pdf diakses 8 Mei 2016 pukul 21.03

[2] https://icpc.baylor.edu/ diakses 8 Mei 2016 pukul 20.51

[3] Munir, Rinaldi. Diktat Kuliah IF2211 Strategi Algoritma. Program Studi Teknik Informatika STEI 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, 8 Mei 2016

Christian Anthony Setyawan

13514085