Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Laporan Tugas Kecil 2 IF2211 Strategi Algoritma

Semester II Tahun 2020/2021

**Penyusunan Rencana Kuliah dengan *Topological Sort***

**(Penerapan *Decrease* *and* *Conquer*)**

Nama : Muhammad Azhar Faturahman

NIM : 13519020

Kelas : K01

**Pendahuluan**

*Decrease and conquer* adalah metode perancangan algoritma dengan mereduksi persoalan menjadi dua upa-persoalan (*sub-problem*) yang lebih kecil, tetapi selanjutnya hanya memproses satu sub-persoalan saja. Berbeda dengan *divide and conquer* yang memproses semua upa-persoalan dan menggabung semua solusi setiap sub-persoalan. Di dalam literatur lama, semua algoritma yang membagi persoalan menjadi dua upa-persoalan yang lebih kecil dimasukkan ke kategori *divide and conquer*. Meskipun demikian, tidak kedua upa persoalan hasil pembagian diselesaikan. Jika hanya satu upa-persoalan yang diselesaikan, maka tidak tepat dimasukkan sebagai algoritma *divide and conquer*. Mereka dikategorikan sebagai *decrease and conquer*.

Algoritma *decrease and conquer* terdiri dari dua tahapan :

1. Decrease : Mereduksi persoalan menjadi beberapa persoalan yang lebih kecil biasanya dua upa-persoalan.
2. Conquer : Memproses satu upa-persoalan secara rekursif atau iteratif.

Tidak ada tahap combine dalam *decrease and conquer*, karena hanya satu upa-persoalan yang diselesaikan.

Tiga varian *decrease and conquer* :

1. *Decrease by a constant*, ukuran instans persoalan direduksi sebesar konstanta yang sama setiap iterasi algoritma.
2. *Decrease by a constant factor*, ukuran instans persoalan direduksi sebesar faktor konstanta yang sama setiap iterasi algoritma.
3. *Decrease by a variable size* : ukuran instans persoalan direduksi bervariasi pada setiap iterasi algoritma.

**Algoritma *Topological Sort* dan kaitannya dengan pendekatan *Decrease and Conquer***

*Topological Sort* adalah suatu metode pengurutan pada graf berarah sehingga didapatkan suatu keterurutan partial linier dari setiap simpulnya yang mana untuk setiap sisi *uv* dari simpul *u* ke simpul *v*, simpul *u* selalu berada di urutan terdepan daripada simpul *v*. *Topological Sort* hanya bisa dilakukan pada suatu graf berarah jika graf berarah tersebut tidak memiliki *directed cycle* (siklus), graf yang tidak memiliki *directed cycle* disebut sebagai *directed acyclic graph* (DAG).

Salah satu pemanfaatan *Topological Sort* adalah dalam merancang suatu penjadwalan, misalnya pada kasus ini menyusun rencana kuliah yang mana pada sebuah mata kuliah terdapat *prerequisite* yang harus terlebih dahulu diambil.

Salah satu algoritma untuk menyelesaikan *Topological Sort* adalah Algoritma Kahn, berikut *pseudocode* algoritma tersebut,

|  |
| --- |
| L ← Empty list that will contain the sorted elements  S ← Set of all nodes with no incoming edge  while S is not empty do  remove a node n from S  add n to L  for each node m with an edge e from n to m do  remove edge e from the graph  if m has no other incoming edges then  insert m into S  if graph has edges then  return error (graph has at least one cycle)  else  return L (a topologically sorted order) |

**Algoritma Kahn (dikutip dari** [**https://en.wikipedia.org/wiki/Topological\_sorting**](https://en.wikipedia.org/wiki/Topological_sorting)**)**

Pada persoalan ini, penulis mengikuti ide dari Algoritma Kahn untuk menyelesaikan permasalahan penyusunan rencana kuliah. Algortima ini dapat digolongkan ke dalam jenis strategi algoritma *decrease and conquer* karena pada setiap langkah iterasi, terdapat penghapusan simpul dari graf (tahap *decrease*) atau dalam hal ini membagi permasalah menjadi dua upa-persoalan, simpul yang dihapus dari graf akan dimasukkan ke dalam himpunan solusi, sedangkan graf yang tersisa akan diproses dengan cara yang sama dengan metode iteratif hingga didapatkan graf kosong (tahap *conquer*) dan proses *topological sort* selesai.

**Source Program**

1. Metode Iteratif

Ditulis menggunakan bahasa pemograman Python 3

|  |
| --- |
| # Nama : Muhammad Azhar Faturahman  # NIM : 13519020  # Kelas : K1  # TUGAS KECIL 1  # IF2211 STRATEGI ALGORITMA  # --- TOPOPGRAPICAL SORT --- #  # Implementasi graph menggunakan adjacent list  ## LIBRARY  ## KAMUS GLOBAL  graphMatkul = {}  urutanMatkul = []  daftarSemester = ["I","II","III","IV","V","VI","VII","VIII","IX","X","XI","XII","XIII","XIV"] # Hardcode it's de way, lebih dari ini DO!  ## FUNGSI DAN MODUL  # Membaca input berupa file, mengembalikan waktu dimulainya program  def bacaFile(namaFile) :  global graphMatkul    try:  alamatFile = "../test/" + namaFile  f = open(alamatFile,'r')    lines = f.readlines()  # print(lines)    i = 0  while (i < len(lines)):  clearLine = lines[i].replace("\n","")  clearLine = clearLine.replace(".","")  clearLine = clearLine.replace(", ",",")  splitLine = clearLine.split(",")    # Isi dictionary dengan keys splitLine[0] dan value splitLine[1:]  if (len(splitLine) > 1):  graphMatkul.update({splitLine[0]: splitLine[1:]})  else:  graphMatkul.update({splitLine[0]: []})    i += 1    f.close()  except FileNotFoundError:  print("\nError: File tidak ditemukan")  # Mengurutkan graph berdasarkan topograpical sort  def topSort():  global graphMatkul, urutanMatkul    # Dicopy dulu biar ga ngeubah yang global bray  olahMatkul = {key: value[:] for key, value in graphMatkul.items()}    # Looping selama olahMatkul belum kosong, WARNING pastikan graph adalah DAC  while (len(olahMatkul) > 0):  found = False    # Cari matkul yang sudah tidak punya prerequisite  for namaMatkul in olahMatkul:  if len(olahMatkul[namaMatkul]) == 0:  urutanMatkul.append(namaMatkul) # Tambah ke list matkul terurut  olahMatkul.pop(namaMatkul) # Hapus dari graph  found = True  break    if found:  # Iterasi untuk hapus namaMatkul tersebut dari matkulLain  for matkulLain in olahMatkul:  if namaMatkul in olahMatkul[matkulLain]:  # Note : olahMatkul[matkulLain] akan menghasilkan list prerequisite maktulLain tersebut  olahMatkul[matkulLain].remove(namaMatkul)    # Iterasi, ulang lagi dari awal untuk graph yang telah dikurangi (Decrease and Conquare)  else:  raise Exception("GraphError: Contains a cycle")    # Menampilkan hasil urutan pada layar  def outputHasil(): # Something I found : Sebenernya ini juga udah bisa toposort wkwkw, tinggal ubah metode looping matkul  global graphMatkul, urutanMatkul, daftarSemester    # listMatkul = list(graphMatkul.keys()).copy()  listMatkul = urutanMatkul.copy()  pastSemester = []  semester = 1    while (len(pastSemester) != len(graphMatkul)):  currSemester = []  i = 0 # Iterator setiap matkul  i = 0 # Iterator setiap matkul  j = 0 # Penanda matkul terakhir yang dicek    while(i < len(listMatkul)): # Iterasi semua matkul yang masih tersedia  matkul = listMatkul[i]  prerequisite = graphMatkul[matkul].copy()    valid = True  for preMatkul in prerequisite: # Iterasi matkul prerequisite  # Cek apakah sudah diambil di semester sebelumnya  if preMatkul not in pastSemester:  valid = False  break  # Cek apakah ada prerequisite diambil bersamaan  elif preMatkul in currSemester:  valid = False  break    if valid: # Tambah ke semester sekarang dan hapus dari matkul tersedia  currSemester.append(matkul)  listMatkul.remove(matkul)  i = j # listMatkul sudah dimodifikasi, harus diulang dari cek terakhir  else:  i += 1  j = i # Biar ga ngulang lagi loopnya dari awal    if (semester-1 < len(daftarSemester)):  print(("Semester "+daftarSemester[semester-1]).ljust(13),end=": ")  for i in range(len(currSemester)):  if (i != len(currSemester) - 1): # Bukan matkul terakhir disuatu semester  print(currSemester[i], end=", ")  else:  print(currSemester[i])    pastSemester = pastSemester + currSemester  semester = semester + 1  else:  print("-- DO --")  break      ## MAIN PROGRAM    def main():  global graphMatkul, urutanMatkul    print()  print("--------------------------------------------")  print(" --- TOPOPGRAPICAL SORT ---")  print("--------------------------------------------")  print("Created by : M Azhar Faturahman (13519020)")  print()  namaFile = input("Masukkan nama file: ")    bacaFile(namaFile)  topSort()    print()  print("--------------------------------------------")  print(" --- HASIL SORTING ---")  print("--------------------------------------------")  # print(urutanMatkul)  outputHasil()  input("\nPress Enter to Continue...")    main() |

1. Metode Rekursif

Ditulis menggunakan bahasa pemograman Python 3

|  |
| --- |
| # Nama : Muhammad Azhar Faturahman  # NIM : 13519020  # Kelas : K1  # TUGAS KECIL 1  # IF2211 STRATEGI ALGORITMA  # --- TOPOPGRAPICAL SORT --- #  # Implementasi graph menggunakan adjacent list  ## LIBRARY  ## KAMUS GLOBAL  graphMatkul = {}  urutanMatkul = []  daftarSemester = ["I","II","III","IV","V","VI","VII","VIII","IX","X","XI","XII","XIII","XIV"] # Hardcode it's de way, lebih dari ini DO!  ## FUNGSI DAN MODUL  # Membaca input berupa file, mengembalikan waktu dimulainya program  def bacaFile(namaFile) :  global graphMatkul    try:  alamatFile = "../test/" + namaFile  f = open(alamatFile,'r')    lines = f.readlines()  # print(lines)    i = 0  while (i < len(lines)):  clearLine = lines[i].replace("\n","")  clearLine = clearLine.replace(".","")  clearLine = clearLine.replace(", ",",")  splitLine = clearLine.split(",")    # Isi dictionary dengan keys splitLine[0] dan value splitLine[1:]  if (len(splitLine) > 1):  graphMatkul.update({splitLine[0]: splitLine[1:]})  else:  graphMatkul.update({splitLine[0]: []})    i += 1    f.close()  except FileNotFoundError:  print("\nError: File tidak ditemukan")  # Mengurutkan graph berdasarkan topograpical sort  def topSort(olahMatkul):  global graphMatkul, urutanMatkul    if len(olahMatkul) == 0: # BASIS  return 0 # DO NOTHING  else: # REKURENS  found = False  # Cari matkul yang sudah tidak punya prerequisite  for namaMatkul in olahMatkul:  if len(olahMatkul[namaMatkul]) == 0: # Jika matkul ini sudah tidak punya prerequisite  found = True  break    if found:  urutanMatkul.append(namaMatkul) # Tambah ke list matkul terurut  olahMatkul.pop(namaMatkul) # Hapus dari graph    # Iterasi untuk hapus namaMatkul tersebut dari matkulLain  for matkulLain in olahMatkul:  if namaMatkul in olahMatkul[matkulLain]:  # Note : olahMatkul[matkulLain] akan menghasilkan list prerequisite maktulLain tersebut  olahMatkul[matkulLain].remove(namaMatkul)    topSort(olahMatkul) # Rekursi, ulang lagi dari awal untuk graph yang telah dikurangi (Decrease and Conquare)  else:  raise Exception("GraphError: Contains a cycle")      # Menampilkan hasil urutan pada layar  def outputHasil(): # Something I found : Sebenernya ini juga udah bisa toposort wkwkw, tinggal ubah metode looping matkul  global graphMatkul, urutanMatkul, daftarSemester    # listMatkul = list(graphMatkul.keys()).copy()  listMatkul = urutanMatkul.copy()  pastSemester = []  semester = 1    while (len(pastSemester) != len(graphMatkul)):  currSemester = []  i = 0 # Iterator setiap matkul  i = 0 # Iterator setiap matkul  j = 0 # Penanda matkul terakhir yang dicek    while(i < len(listMatkul)): # Iterasi semua matkul yang masih tersedia  matkul = listMatkul[i]  prerequisite = graphMatkul[matkul].copy()    valid = True  for preMatkul in prerequisite: # Iterasi matkul prerequisite  # Cek apakah sudah diambil di semester sebelumnya  if preMatkul not in pastSemester:  valid = False  break  # Cek apakah ada prerequisite diambil bersamaan  elif preMatkul in currSemester:  valid = False  break    if valid: # Tambah ke semester sekarang dan hapus dari matkul tersedia  currSemester.append(matkul)  listMatkul.remove(matkul)  i = j # listMatkul sudah dimodifikasi, harus diulang dari cek terakhir  else:  i += 1  j = i # Biar ga ngulang lagi loopnya dari awal    if (semester-1 < len(daftarSemester)):  print(("Semester "+daftarSemester[semester-1]).ljust(13),end=": ")  for i in range(len(currSemester)):  if (i != len(currSemester) - 1): # Bukan matkul terakhir disuatu semester  print(currSemester[i], end=", ")  else:  print(currSemester[i])    pastSemester = pastSemester + currSemester  semester = semester + 1  else:  print("-- DO --")  break      # MAIN PROGRAM    def main():  global graphMatkul, urutanMatkul    print()  print("--------------------------------------------")  print(" --- TOPOPGRAPICAL SORT ---")  print("--------------------------------------------")  print("Created by : M Azhar Faturahman (13519020)")  print()  namaFile = input("Masukkan nama file: ")    bacaFile(namaFile)  olahMatkul = {key: value[:] for key, value in graphMatkul.items()} # Dicopy dulu biar ga ngeubah yang global bray  topSort(olahMatkul)    print()  print("--------------------------------------------")  print(" --- HASIL SORTING ---")  print("--------------------------------------------")  # print(urutanMatkul)  outputHasil()  input("\nPress Enter to Continue...")    main() |

***Screenshot* Hasil (Input dan Output)**

|  |  |  |
| --- | --- | --- |
| No | Input | Output |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  | Pada kasus ini, penyusunan rencana kuliah hanya bisa sampai semester 14, sehingga jika lebih dari 14 semester akan DO. |
| 9 |  | Pada kasus ini akan terjadi error karena dalam graf terdapat *directed cycle* sehingga tidak dapat dilakukan *topological sort.* |

**Alamat Drive**

<https://drive.google.com/drive/folders/1Dm16rActnP-3_u66P86Sn1Iw_88fr5zZ?usp=sharing>

**Cek List Program**

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi | ✓ |  |
| 1. Program berhasil *running* | ✓ |  |
| 1. Program dapat menerima berkas input dan menuliskan output. | ✓ |  |
| 1. Luaran sudah benar untuk semua kasus input. | ✓ |  |

**Referensi**

1. R. Munir, “Diktat Kuliah Strategi Algoritma ITB (IF2211)”, 2020, <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/stima20-21.htm> diakses pada 27 Februari 2021.
2. <https://en.wikipedia.org/wiki/Topological_sorting> diakses pada 27 Februari 2021.
3. <https://www.youtube.com/watch?v=eL-KzMXSXXI> diakses pada 27 Februari 2021.
4. <https://www.w3schools.com/python/python_dictionaries.asp> diakses pada 27 Februari 2021.