Курсови проекти (част I)
за домашна работа
МДГП, 2022

Проектите описани долу заместват писмено контролно изпитване в контекста на дистанционно обучение.

Изберете едно измежду следните:

1. Прости числа и техните прости фактори
2. Графи на Кейли и групи на пермутациите
3. Симулация на случайни мрежи
4. Презентация върху научна публикация

Групи: Индивидуално, двама или трима души
Краен срок: 4 Декември 2022, 23:59
Език на имплементация за 1, 2 и 3: по избор / няма ограничения

За първите три варианта е даден псевдо-код (който улеснява задачата); статиите за вариант 4 ще бъдат качени в Moodle и/или изпратени на всеки по електронна поща. Независимо кой от вариантите изберете, трябва да имате готовност да презентирате/демонстрирате решението и резултатите си, а след това да отговаряте на въпроси по темата. За визуализация на графите от 1, 2 или 3 ползвайте софтуера Gephi, заедно с gexf формат - студентите следва да демонстрират работещата програма, и да обяснят своя сорс код, както и получените резултати.

Описание на данните на изхода:
Имплементираната програма трябва да поддържа функционалността за експортиране на генерираните графи в GEXF формат, който описва типа граф, множеството от върховете и техните обозначения, както и ребрата (ID на върха-начало и ID на върха-край). Единствените видове информация нужни за този тази цел може да видите в долния пример:

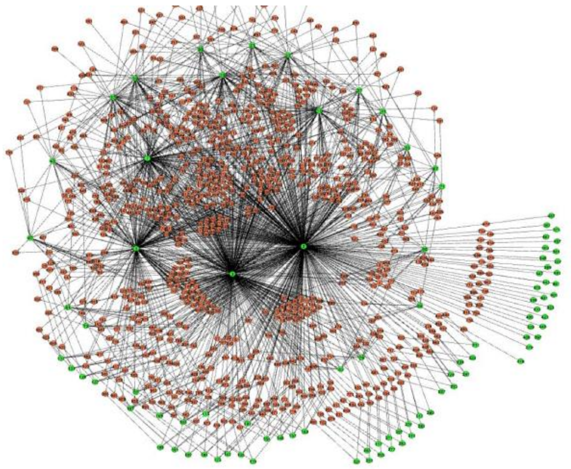
```xml
<?xml version="1.0" encoding="UTF-8"?>
<gexf xmlns="http://www.gexf.net/1.3" version="1.3" xmlns:viz="http://www.gexf.net/1.3/viz" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.gexf.net/1.3http://www.gexf.net/1.3/gexf.xsd">
  <graph defaultedgetype="directed" mode="static">
    <nodes>
      <node id="0" label="Label A">
      </node>
      <node id="1" label="Label B">
      </node>
      …
      <node id="123" label="Label N">
      </node>
    </nodes>
    <edges>
      <edge id="0" source="0" target="12"></edge>
      <edge id="1" source="100" target="61"></edge>
      <edge id="2" source="23" target="97"></edge>
      …
      <edge id="50" source="77" target="105"></edge>
      <edge id="51" source="97" target="123"></edge>
    </edges>
  </graph>
</gexf>
```

След като генерирате XML представянето на графите, уверете се, че Gephi може успешно да отвори получените файлове (в противен случай има грешка, която трябва да се коригира).

Експортирайте визуализираните с помощта на Gephi графи, както е описано по-горе, в PNG формат.

1. Цели числа и техните прости фактори

Вземаме предвид, че всяко не-просто число n можем да представим като произведение на по-малки от него прости числа, както следва: $n = p_1 p_2 \dots p_k$. Вижте реф. [1], [2] за повече информация. Нека построим граф, който показва числата до 1000 като върхове, и оцветява простите числа в различен цвят. Ребрата свързват дадено не-просто число със всичките му прости фактори (които, разбира се, трябва да бъдат открити/пресметнати при построяването на графа). При нужда, потърсете допълнителни теоретични материали или примери в интернет.



Фигура 1: *Целите числа от 2 до 1000 и техните прости фактори (оцветени в зелено).*

Псевдо-кода за получаване на графа от горната фигура е описан в *Алгоритъм 1*.

Алгоритъм 1:

```
inputs: N //an integer (e.g. 1000)
init:
  g ← a graph with nodes {1, …, N} and empty edge set, default node colour attribute: 'brown'

function iterate(to_num):
  for i ∊ {1, …, to_num}:
    is_prime ← is_prime_number(i)  //check whether i is prime
    factors_list ← get_prime_factors(i)  //calculate all prime factors of i (Trial division or other method)
    if is_prime is True:
      g.nodes[i].set_colour('green')  //update g: change default node colour
    else:
      for factor_num ∊ factors_list:
        g.create_directed_edge(factor_num, i)  // update g: create an edge from factor_num to i

// run it:
iterate(N)
to_xml(g)  // convert the graph into XML format
```

Референции:

1. https://en.wikipedia.org/wiki/Fundamental_theorem_of_arithmetic
2. https://mathworld.wolfram.com/FundamentalTheoremofArithmetic.html

2. Графи на Кейли и групи на пермутациите

Понятия от алгебрата като Симетрична група и Група на пермутациите, и нейните генератори, са абстрактни и невинаги лесни за разбиране. Въпреки това, графите съответствщи на малки симетрични групи могат лесно да бъдат построени и визуализирани с помощта на прости алгоритми. Целта на това упражнение е да се построи автоматично граф, съответстващ на групата Sym(4), с помощта на *Алгоритъм 2*. За решението е достатъчно да се използва подходяща инициализация и една рекурсивна функция. Върховете в примерната Фигура 2 съответстват на различни пермутации на крайна група от елементи, следователно началния връх/елемент от групата и списъка от пермутации, които да бъдат прилагани, може да бъдат подадени като параметри – ролята на рекурсивната функция ще бъде да приложи последователно всички пермутации, извиквайки себе си след това, като по този начин обходи всички възможни върхове.

Алгоритъм 2:

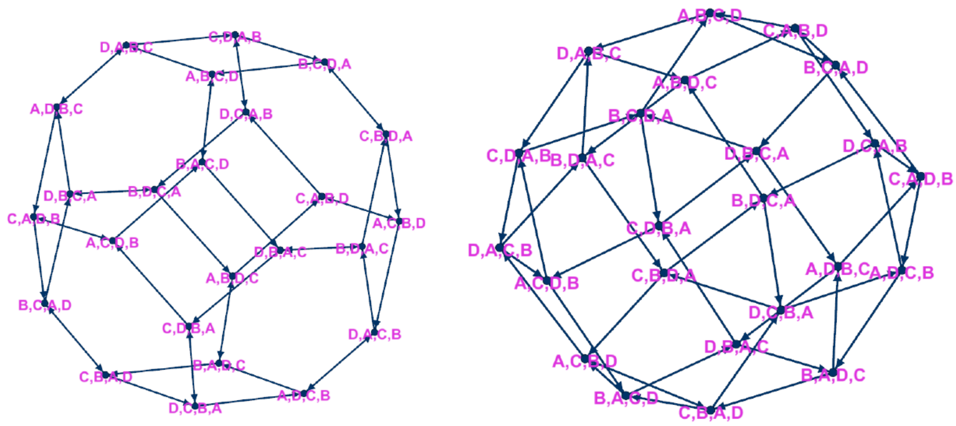
```
inputs:
   permutations ← list of lists   //for example two permutations: [[4,1,2,3], [2,3,1,4]]
   init_node ← [‘A’, ‘B’, ‘C’, ‘D’, …]  //initial permutation
init:
   visited ← set()
   edges ← dictionary()

function recursion(node):
   for perm ∊ permutations:
      new_node ← permute(node, permutation)  //ex: BACD = permute(ABCD, [2,1,3,4])
      if new_node ∊ visited:
         edges.add_edge(node, new_node)  // add a directed edge to the edges dictionary
      else:
         visited.add(new_node)
         edges.add_edge(node, new_node)  // add a directed edge to the edges dictionary
         recursion(new_node)  // recursive call

// run it:
visited.add(init_node)
recursion(init_node)
to_xml(visited, edges)  // convert the graph into XML format
```

Интересно е, че в зависимост от избраните пермутации, полученият граф би имал едни и същи върхове, но различна структура от ребра между тях. Например, с цел представянето на Sym(4) като граф, следните пермутации могат да бъдат приложени:

- P1: ((1, 2, 3, 4), (4, 1, 2, 3)) и P2: ((1, 2, 3, 4), (2, 1, 3, 4)) за да получим графа в лявата половина на Фигура 2
- P1: ((1, 2, 3, 4), (4, 1, 2, 3)) и P3: ((1, 2, 3, 4), (2, 3, 1, 4)), посредство които получаваме графа в дясната половина на Фигура 2

Нека отбележим, че *Алгоритъм 2* е приложим и за групи от пермутации с повече от 4 елемента. В рамиките на този пример, обозначенията на върховете т.е. елементите, върху които ще се прилагат пермутациите, нека бъдат съставени от първите 4 латински букви: например ABCD, DCBA и т.н. Пермутациите (вж. горе) имат за цел да променят позициите на буквите – напр. P1 ще премести четвъртата буква (независимо коя е тя) на първа позиция, първата буква на втора позиция, втората на позиция 3, а третата буква съответно на последната позиция 4.

Указаните горе пермутации са достатъчни за изпълнението на задачата, но разбира се, можете да експериментирате.



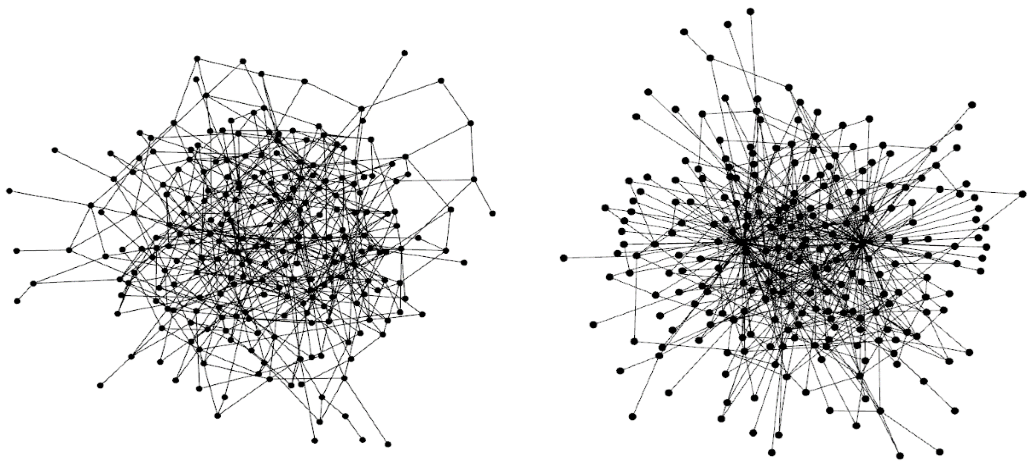
Фигура 2: *Графи съответстващи на Sym(4), генерирани с помощта на Алгоритъм 2. Пермутации: P1 и P2 (вляво); P1 и P3 (вдясно).*

Референции:
1. https://en.wikipedia.org/wiki/Cayley_graph
2. https://mathworld.wolfram.com/PermutationGroup.html
3. https://mathworld.wolfram.com/SymmetricGroup.html

3. Симулиране на случайни мрежи

Поведението на случайните графи (вж. Preferential attachment, Scale-free behaviour [2]) е част както от теорията на графите, така и от вероятностите и статистиката [3], като това е видимо и от алгоритъмът по-долу. Ако графите от предишните примери показват детерминистични структури, то тук моделираме недетерминистично/случайно поведение. В частност, това упражнение разглежда:

- случайни графи според модела на Erdős–Rényi [1], с вероятност за наличие (създаване) на случайно ребро контролирана от параметъра *p*;
- графи с поведение на преференциално прикавчане (preferential attachment) на нови върхове към съществуващи върхове с по голяма степен т.е. повече налични ребра, подобно на модела на Barabási-Albert [2].



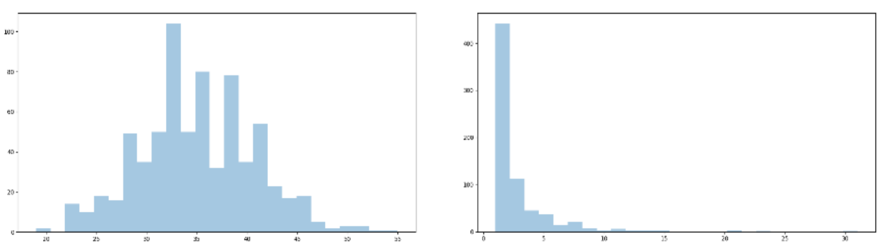
Фигура 3: Случайни графи *генерирани с Алгоритъм 3: Erdős–Rényi модел (вляво) и граф с преференциално прикачване на новите върхове (вдясно)*

В първия случай, *Алгоритъм 3* може да бъде извикан с параметър *preferential = False*; във втория случай, съответно с *preferential = True*, което ще позволи вероятността за създаване на ребра да зависи от текущите степени на върховете.

Алгоритъм 3:

```
inputs:
  p ← number ∈ [0, 1] //probability between 0 and 1
  preferential ← {True, False}  //preferential attachment or Erdős–Rényi model
  pw ← 0.0  //power parameter, equal to 0.0 by default

function random_graph(N, p, preferential, pw=0.0):
  vertices ← {1, …, N}
  edges ← set()
  deg ← [0, 0, …, 0] //vertex degrees initialization - N zeros

  for i ∈ {1, …, N}:
    for j ∈ {1, …, i-1}:
      u ← U([0, 1]) //uniformly distributed random number between 0 and 1
      if preferential = =True:
        p ← max(deg_i, deg_j)^(1+pw) / (1 + Σ_{k=0}^{i} deg_k)
        if deg_j == 0:
          edges.add_edge(j, r) where r is a randomly selected vertex
          deg_j ++; deg_r ++
      if u > 1 − p :
        deg_i ++; deg_j ++
        edges.add_edge(i, j)
  return g, deg

// run it:
g, deg = random_graph(250, 0.2, False, 0.0)
to_xml(g)  // convert the graph into XML format
plot(histogram(deg))
```

В допълнение към визуалното сравняване на мрежите, получени с този алгоритъм, би било добре също да разгледаме емпиричните разпределения на степените на върховете на двата типа графи - вж. следните закони за разпределение на вероятност: Биномно разпределение (Binomial distribution) и Степенно разпределение (Power-law distribution).



Фигура 3.1: *Разпределение на степените на върховете на случайни графи, генерирани с Алгоритъм 3: модел на Erdős–Rényi (вляво) и граф с преференциално прикачване (вдясно)*

Референции:

1. Erdős, P., & Rényi, A. (1960). *On the evolution of random graphs.* Publ. Math. Inst. Hung. Acad. Sci, 5(1), 17-60.
2. Barabási, A. L., & Albert, R. (1999). *Emergence of scaling in random networks.* Science, 286(5439), 509-512.
3. Lovász, L. (2012). *Large networks and graph limits* (Vol. 60). American Mathematical Soc..
4. https://en.wikipedia.org/wiki/Random_graph

4. Презентация върху научна публикация

В допълнение към този документ, ще получите по електронна поща и/или намерите качени в Moodle 20 на брой научни публикации в сферата на теорията на графите и нейните приложения. Публикациите са с различни теми и трудност, като студентите/групите избрали да подготвят презентация върху публикация е достатъчно да изберат една измежду всички. Моля не забравяйте да ми изпратите номера и заглавието на публикацията, която сте избрали.

Презентация с големина до 10 слайда би била достатъчна, като следва да бъде представена в рамките на 10-15 минути. В края на презентацията може да бъдат задавани въпроси, както от преподавателя, така и от останалите студенти. Представяне в Дискорд със споделяне на екран е подходящ вариант за презентиране.