Part9-Graph&The Contraction Algorithm

Completed on	@2020/03/22
Key videos	https://www.coursera.org/learn/algorithms-divide-conquer/lecture/b3CWT/graph-representations https://www.coursera.org/learn/algorithms-divide-conquer/lecture/FKAnq/random-contraction-algorithm https://www.coursera.org/learn/algorithms-divide-conquer/lecture/4TLKM/analysis-of-contraction-algorithm
Note	Application of the randomized algorithms to min cuts problem of the graph algorithms.

Note

- Overview

  Part IX --- GRAPHS AND THE CONTRACTION ALGORITHM. The second set of lectures for this week is a segue from randomized algorithms to graph algorithms. We first review graphs and the most standard ways of representing them (most commonly, by adjacency lists). We then consider the random contraction algorithm, discovered by Karger "only" 20ish years ago (while a PhD student here at Stanford). This algorithm solves the minimum cut problem --- given an undirected graph, separate the vertices into two non-empty groups to minimize the number of "crossing edges". Such problems come up when reasoning about, for example, physical networks, social networks, and images. This algorithm was perhaps the first strong evidence that graph problems could be added to the long list of "killer applications" of random sampling. Don't tune out before the final plot twist --- a simple but useful trick for transforming an algorithm that almost always fails into one that almost always succeeds.

  HOMEWORK: Problem Set #4 has five questions about the randomized selection algorithm, cuts in graphs, and the contraction algorithm. Programming Assignment #4 asks you to implement the contraction algorithm and use it to compute the min cut of the graph that we provide.

▼ Graphs and Minimum Cuts

- 意义：进一步讨论随机算法，并在图上进行应用！

Further practice with randomized algorithms

– In a new application domain (graphs)

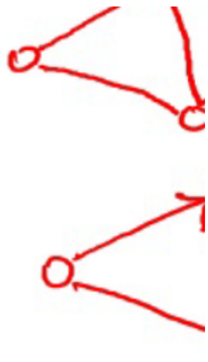
Introduction to graphs and graph algorithms

- 图的基本知识：点与边是两个组成部分

Two ingredients

- Vertices aka nodes (V)
- Edges (E) = pairs of vertices
  - can be undirected [unordered pair]

  or directed [ordered pair] (aka arcs)

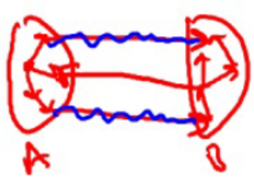


Examples: road networks, the Web, social networks, precedence constraints, etc.

- Cuts of graphs：将图分为两个非空子集的表示

Definition: a cut of a graph (V, E) is a partition of V into two non-empty sets A and B.

[undirected]



[directed]

Definition: the crossing edges of a cut (A, B) are those with:

- the one endpoint in each of (A, B) [undirected]
- tail in A, head in B [directed]

cuts的可能总数：每个点两种可能 ⇒ 2^n

- MIN CUT问题：输入一个图，输出找到一个cut，拥有最少的crossing edges

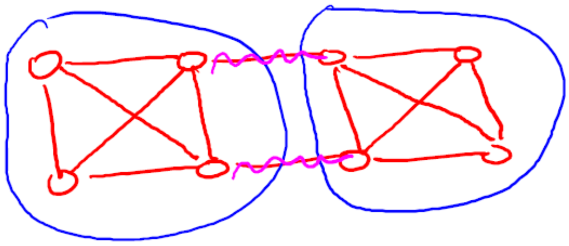
  - INPUT: An undirected graph G = (V, E).
  [Parallel edges allowed]
  [See other video for representation of the input]

  - GOAL: Compute a cut with fewest number of crossing edges. (a min cut)

  min cut问题的例子

  What is the number of edges crossing a minimum cut in the graph shown below?

  - 1
  - 2
  - 3
  - 4

- 相关领域的运用

  - indentify network bottlenecks / weaknesses
  - community detection in social networks
  - image segmentation
    - input = graph of pixels
    - use edge weights

      [(u,v) has large weight ⇔ “expect” u,v to come from some object]

    hope: repeated min cuts identifies the primary objects in picture.

▼ Graph Representations

- 边与点的关系，如果有n个点，那么组成图所需边最少和最多分别是多少?

Consider an undirected graph that has n vertices, no parallel edges, and is connected (i.e., "in one piece"). What is the minimum and maximum number of edges that the graph could have, respectively?

- [x] $n-1$ and $n(n-1)/2$

Correct

- [ ] $n-1$ and n^2
- [ ] n and 2^n

- 稀疏/稠密图

Let <u>n</u> = # of vertices,   <u>m</u> = # of edges.

In most (but not all) applications, m is $\Omega(n)$ and $O(n^2)$

• in a "sparse" graph, m is or is close to $O(n)$
• in a "dense" graph, m is closer to $\theta(n^2)$

- 图的表示方法：

1. 邻接矩阵：A_ij即为边的表示

Represent G by a n x n 0-1 matrix A where
$A_{ij} = 1 \Leftrightarrow$ G has an i-j edge $\textcircled{i}\!\longrightarrow\!\textcircled{j}$

<u>Variants</u>

• A_{ij} = # of i-j edges (if parallel edges)
• A_{ij} = weight of i-j edge (if any)
• $A_{ij} = \begin{cases} +1 \text{ if } \bigcirc\!\!\longrightarrow\!\!\bigcirc \\ -1 \text{ if } \bigcirc\!\!\longleftarrow\!\!\bigcirc \end{cases}$

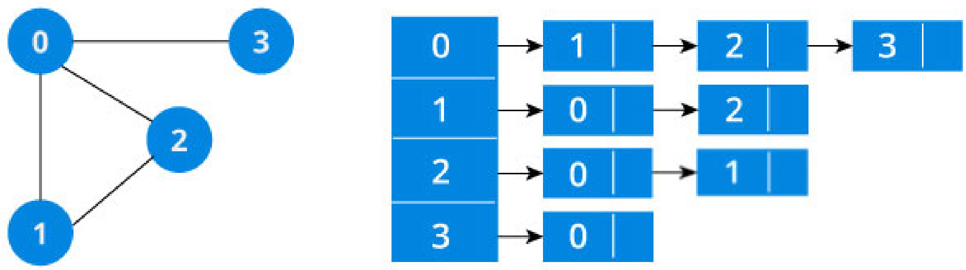
最大所需空间复杂度即为n^2

How much space does an adjacency matrix require, as a function of the number n of vertices and the number m of edges?

- $\theta(n)$
- $\theta(m)$
- $\theta(m+n)$
- $\theta(n^2)$

Correct

2. Adjacency Lists邻接表：



Ingredients	Space
• array (or list) of vertices	$\theta(n)$
• array (or list) of edges	$\theta(m)$
• each edge points to its endpoints	$\theta(m)$
• each vertex points to edges incident on it	$\theta(m)$
	$\theta(m+n)$
	[or $\theta(max\{m,n\})$]

one-to-one correspondence !

Question: which is better?
Answer: depends on graph density and operations needed.
This course: focus on adjacency lists.

▼ Random Contraction Algorithm

- 最小图割问题：输入一个无向图G=(V, E)输出带有crossing edges的最小图割
- Random Contraction Algorithm随机压缩算法步骤 ⇒ 但是可能有失败的机率

While there are more than 2 vertices:
- pick a remaining edge (u,v) uniformly at random
- merge (or "contract") u and v into a single vertex
- remove self-loops

return cut represented by final 2 vertices.

▼ Analysis of Contraction Algorithm

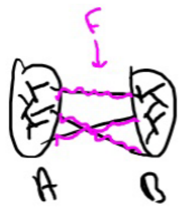
- 基本假设：min cut有k条crossing edges ⇒ 集合F

<u>Question</u>: what is the probability of success?
Fix a graph G = (V, E) with n vertices, m edges.
Fix a minimum cut (A, B).
Let k = # of edges crossing (A, B). (Call these edges F)

- 失败条件：压缩到所需的k条crossing edges ⇒ 成功概率则是不压缩到集合F的概率

  最终成功的概率即为压缩到只剩下两条边并仍然输出(A,B)的概率

1. Suppose an edge of F is contracted at some point
   ⇨ algorithm <u>will not</u> output (A,B).



2. Suppose only edges inside A or inside B get contracted ⇨ algorithm <u>will</u> output (A, B).

<u>Thus</u>: Pr [output is (A, B)] = Pr [never contracts an edge of F]

Let S_i = event that an edge of F contracted in iteration i.
<u>Goal</u>: Compute $\Pr[\neg S_1 \wedge \neg S_2 \wedge \neg S_3 \wedge \ldots\ldots \wedge \neg S_{n-2}]$

压缩到概率为k/m

What is the probability that an edge crossing the minimum cut (A, B) is chosen in the first iteration (as a function of the number of vertices n, the number of edges m, and the number k of crossing edges)?

- ○ k/n
- ○ k/m
- ○ k/n^2
- ○ n/m

$$\Pr[S_1] = \frac{\#\text{ of crossing edges}}{\#\text{ of edges}} = \frac{k}{m}$$

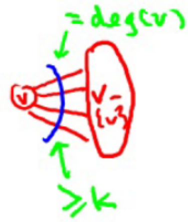
- 第一次迭代：假设每个点至少有k个自由度；此外，所有点的自由度集合自由度为2m（两倍于边的总数）≥ kn ⇒ m ≥ kn/2

  因此，结合之前结论推出，第一次迭代压缩导致最终失败的概率：

  P(S1) = k/m ≤ 2/n

Key Observation: $\underbrace{\text{degree}}_{\#\text{ of incident edges}}$ of each vertex is at least k

Reason: each vertex v defines a cut ({v}. V-{v}).



Since $\underbrace{\sum_v degree(v)}_{\geq kn} = 2m$, we have $m \geq \frac{kn}{2}$

Since $\Pr[S_1] = \frac{k}{m}$, $\Pr[S_1] \leq \frac{2}{n}$

- 第二次迭代：考虑条件概率！

  P(-S1) ≥ 1 - 2/n；P(-S2|-S1) = 1 - k/(m-1) ≥ 1 - 2/(n-1)

$$\underline{\text{Recall}}:\ \Pr[\neg S_1 \wedge \neg S_2] = \underbrace{\Pr[\neg S_2|\neg S_1]}_{= 1 - \frac{k}{\#\text{ of remaining edge}}} . \underbrace{\Pr[\neg S_1]}_{\geq (1-\frac{2}{n})}$$

what is this?

Note: all nodes in contracted graph define cuts in G (with at least k crossing edges).

➢ all degrees in contracted graph are at least k

So: # of remaining ec $\geq \frac{1}{2}k(n-1)$

So $\Pr[\neg S_2|\neg S_1] \geq 1 - \dfrac{2}{(n-1)}$

- 所有迭代

<u>In general</u>:

$$\Pr[\neg S_1 \wedge \neg S_2 \wedge \neg S_3 \wedge .... \wedge \neg S_{n-2}]$$
$$= \underline{\Pr[\neg S_1]}\ \underline{\Pr[\neg S_2|\neg S_1]}\ \Pr[\neg S_3|\neg S_2 \wedge \neg S_1] ......\Pr[\neg S_{n-2}|\neg S_1 \wedge ... \wedge \neg S_{n-3}]$$
$$\geq (1-\tfrac{2}{n})(1-\tfrac{2}{n-1})(1-\tfrac{2}{n-2}).....(1-\tfrac{2}{n-(n-4)})(1-\tfrac{2}{n-(n-3)})$$
$$= \frac{\cancel{n-2}}{n} \cdot \frac{\cancel{n-3}}{n-1} \cdot \frac{\cancel{n-4}}{\cancel{n-2}} ...... \frac{2}{\cancel{4}} \cdot \frac{1}{\cancel{3}} = \frac{2}{n(n-1)} \geq \frac{1}{n^2}$$

<u>Problem</u>: <u>low</u> success probability! (<u>But</u>: non trivial)

recall $\simeq 2^n$ cuts !

$$\Pr[\neg S_1 \wedge \neg S_2 \wedge \neg S_3 \wedge \ldots \wedge \neg S_{n-2}] = \frac{2}{n-1} \geq \frac{1}{n^2}$$

相比于2^n指数级的可能，整体的效果仍然十分可观！

- 重复的尝试

<u>Solution</u>: run the basic algorithm a large number N times, remember the smallest cut found.

<u>Question</u>: how many trials needed?

Let T_i = event that the cut (A, B) is found on the i^{th} try.

➢ by definition, different T_i's are independent

<u>So</u>: $\Pr[\text{all N trails fail}] = \Pr[\neg T_1 \wedge \neg T_2 \wedge \ldots \wedge \neg T_N]$

$$= \prod_{i=1}^{N} \Pr[\neg T_i] \leq (1 - \frac{1}{n^2})^N$$

By independence !

由于，在第i次尝试找到的可能性都是独立的，因此总体概率可以直接按照乘法原理得到

<u>Calculus fact:</u> ∀real numbers x, $1+x \leq e^x$

$\Pr[\text{all trials fail}] \leq (1 - \frac{1}{n^2})^N$

<u>So</u>: if we take $N = n^2, \Pr[\text{all fail}] \leq (e^{-\frac{1}{n^2}})^{n^2} = \frac{1}{e}$

If we take $N = n^2 \ln n, \Pr[\text{all fail}] \leq (\frac{1}{e})^{\ln n} = \frac{1}{n}$

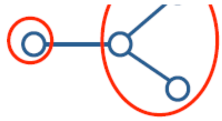
<u>Running time</u>: polynomial in n and m but slow $(\Omega(n^2 m))$

But: can get big speed ups (to roughly $O(n^2)$) with more ideas.

最后部分还需要更深的理解！

▼ Counting Minimum Cuts

- 计算Min Cut的个数



NOTE: A graph can have multiple min cuts.
[e.g., a tree with n vertices has (n-1) minimum cuts]

QUESTION: What's the largest number of min cuts that a graph with n vertices can have?

ANSWER: $\binom{n}{2} = \frac{n(n-1)}{2}$

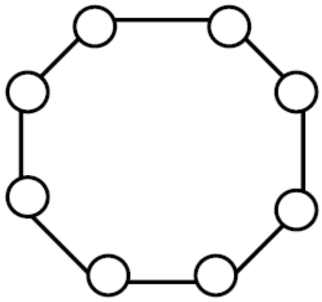
- Lower bound下界

  n-cycle所有的分类都是Min Cut

Consider the n-cycle.

NOTE: Each pair of the n edges defines a distinct minimum cut (with two crossing edges).

➢ has $\geq \binom{n}{2}$ min cuts



(n = 8)

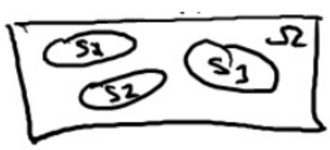
- Upper bound上界

Let (A_1, B_1), (A_2, B_2), …, (A_t, B_t) be the min cuts of a graph with n vertices.
By the Contraction Algorithm analysis (without repeated trials):

$$\Pr[\underbrace{output = (A_i, B_i)}_{S_i}] \geq \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}} \quad \forall i = 1, 2, .., t$$

Note: S_i's are disjoint events (i.e., only one can happen)
➢ their probabilities sum to at most 1

Thus: $\frac{t}{\binom{n}{2}} \leq 1 \Rightarrow t \leq \binom{n}{2}$



References

- Graph (abstract data type)

https://en.wikipedia.org/wiki/Graph_(abstract_data_type)

- Minimum cut

[Minimum cut](https://en.wikipedia.org/wiki/Minimum_cut)
https://en.wikipedia.org/wiki/Minimum_cut

https://www.geeksforgeeks.org/kargers-algorithm-for-minimum-cut-set-1-introduction-and-implementation/

- Karger's algorithm

[Karger's algorithm](https://en.wikipedia.org/wiki/Karger%27s_algorithm)
https://en.wikipedia.org/wiki/Karger%27s_algorithm

[Edge contraction](https://en.wikipedia.org/wiki/Edge_contraction)
https://en.wikipedia.org/wiki/Edge_contraction

- Adjacency Lists

[Adjacency list](https://en.wikipedia.org/wiki/Adjacency_list)
https://en.wikipedia.org/wiki/Adjacency_list

[Adjacency List](https://www.programiz.com/dsa/graph-adjacency-list)
https://www.programiz.com/dsa/graph-adjacency-list

图示与代码实现(python)：

```
graph = {'A': set(['B', 'C']),
         'B': set(['A', 'D', 'E']),
         'C': set(['A', 'F']),
         'D': set(['B']),
         'E': set(['B', 'F']),
         'F': set(['C', 'E'])}
```