

Summary: Lecture 10

Summary for the chapters 11.1 up to page 245 and 11.2 (page 258 optional). [5, 1]

Randomized algorithms

Algorithms based on randomization.

(The algorithm employs a degree of randomness as part of its logic or procedure.)

Bipartite matching

Bipartite Graph

A graph $G = (U, V, E)$ is called bipartite if the vertices can be divided into two disjoint and independent sets U and V . (There are no edges between two elements of U or two elements of V).

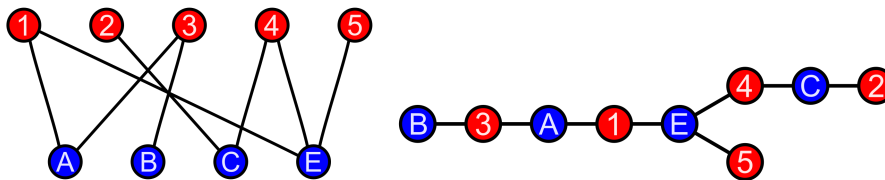


Figure 1: Examples of bipartite graphs with U and V marked in red and blue [2]

Problem: BipartiteMatching

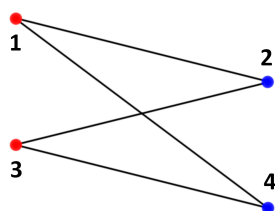
Given: Bipartite graph $G = (U, V, E)$.

Is there a perfect matching $M \subseteq E$ such that for any two edges (u, v) and (u', v') in M $u \neq u'$ and $v \neq v'$.

In other words: A matching in a Bipartite Graph is a set of the edges chosen in such a way that no two edges share an endpoint. The matching M is called perfect if for every node in V there is some edge in M . [3, 5, 4]

- construct bipartite graph with n nodes as $n \times n$ matrix A
- the element $A_{i,j}$ is a variable $x_{i,j}$ if $(i, j) \in E$
- the element $A_{i,j}$ is 0 if $(i, j) \notin E$

Example:



$$A = \begin{pmatrix} 0 & x_{1,2} & 0 & x_{1,4} \\ x_{2,1} & 0 & x_{2,3} & 0 \\ 0 & x_{3,2} & 0 & x_{3,4} \\ x_{4,1} & 0 & x_{4,3} & 0 \end{pmatrix}$$

Questions:

Is EVERY node contained in the subset M when it is a perfect matching? Does a perfect matching then only exist with an even number of vertices and $|U| = |V|$?

Determinant calculation

Leibniz-formula:

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \quad |A| = a \cdot e \cdot i + b \cdot f \cdot g + c \cdot d \cdot h - g \cdot e \cdot c - h \cdot f \cdot a - i \cdot d \cdot b$$

$$B = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad |B| = 1 \cdot 5 \cdot 9 + 2 \cdot 6 \cdot 7 + 3 \cdot 4 \cdot 8 - 7 \cdot 5 \cdot 3 - 8 \cdot 6 \cdot 1 - 9 \cdot 4 \cdot 2 = 0$$

$$|A| = \sum_{\pi} \sigma(\pi) \prod_{i=1}^n A_{i,\pi(i)}$$

- $\sigma(\pi)$ decides if + or -
- leads to $n!$ summands
- Example: $n = 3$
 $3! = 6$ summands
 6 permutations for π
 $+1:$ $(1 \ 2 \ 3) \ (2 \ 3 \ 1) \ (3 \ 1 \ 2)$
 $-1:$ $(3 \ 2 \ 1) \ (2 \ 1 \ 3) \ (1 \ 3 \ 2)$
 $\rightarrow |A| = A_{1,1} \cdot A_{2,2} \cdot A_{3,3} + A_{2,1} \cdot A_{3,2} \cdot A_{1,3} + \dots$

Gaussian elimination:

- Gauß algorithm for solving LSE (linear systems of equations)
- allowed operations:
 - addition of rows
 - subtraction of rows
 - multiply row with integer x
 - divide row by integer x
 - switch to rows
- wanted: upper triangular form (all entries below the diagonal 0)
- determinant is the product of the diagonal entries
- Example:

$$\begin{pmatrix} 1 & 3 & 2 & 5 \\ 1 & 7 & -2 & 4 \\ -1 & -3 & -2 & 2 \\ 0 & 1 & 6 & 2 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 3 & 2 & 5 \\ 0 & 4 & -4 & -1 \\ 0 & 0 & 0 & 7 \\ 0 & 1 & 6 & 2 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 3 & 2 & 5 \\ 0 & 4 & -4 & -1 \\ 0 & 0 & 0 & 7 \\ 0 & 0 & 7 & 2\frac{1}{4} \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 3 & 2 & 5 \\ 0 & 4 & -4 & -1 \\ 0 & 0 & 7 & 2\frac{1}{4} \\ 0 & 0 & 0 & 7 \end{pmatrix}$$

Figure 2: Examples gaussian elimination [5]

$$|A| = 1 \cdot 4 \cdot 7 \cdot 7 = 196$$

Symbolic Determinants

Symbolic matrix:

- matrix with variables instead of numerical entries
- Example:

$$\begin{pmatrix} x & w & z \\ z & x & w \\ y & z & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} x & w & z \\ 0 & \frac{x^2-zw}{x} & \frac{wx-z^2}{x} \\ 0 & \frac{zx-wy}{x} & -\frac{zy}{x} \end{pmatrix} \Rightarrow \begin{pmatrix} x & w & z \\ 0 & \frac{x^2-zw}{x} & \frac{wx-z^2}{x} \\ 0 & 0 & -\frac{yz(xz-xw)+(zx-wy)(wx-z^2)}{x(x^2-zw)} \end{pmatrix}$$

Figure 3: Examples gaussian elimination with symbolic matrix [5]

- subdeterminants have exponentially many terms
- wanted result: know if determinant is 0

Monte Carlo algorithm

Check if determinant of symbolic matrix is 0:

- use arbitrary integers for the variables
→ numerical matrix
- calculate determinant of numerical matrix:
 - if not 0:
determinant of symbolic matrix is not 0
 - if 0:
determinant of symbolic matrix is probably 0
 - * numbers could be chosen such that the numerical determinant is 0 even though the symbolic one is not 0

Monte Carlo algorithm

Randomized algorithm for deciding if a graph G has a perfect matching with calculating the determinant of the corresponding matrix A to G .

- choose m random integers i_1, \dots, i_m between 0 and $2m$
- compute the determinant $|A|(i_1, \dots, i_m)$ with the Gaussian elimination
- if $|A|(i_1, \dots, i_m) \neq 0$ reply G has a perfect matching
- if $|A|(i_1, \dots, i_m) = 0$ reply G has probably no perfect matching

- if perfect matching found: decision is reliable and final
- if perfect matching not found: possibility of false negative

Monte Carlo algorithm

(Algorithm above) decides whether a symbolic matrix is **not** indentially to zero.

Reducing chance of false negatives:

- perform many independent experiments
- chose each time random integers (independently)
- repeat k times the evaluation of the determinant of the symbolic matrix
 - answer always zero:
chance that G has no perfect matching is higher $(1 - (\frac{1}{2})^k)$
 - answer different from zero once:
perfect matching exists

Monte Carlo algorithm:

- Monte Carlo algorithm has no false positives
- probability of false negatives is bounded
- time needed always polynomial

Randomized complexity classes

Monte Carlo Turing Machine

A polynomial Monte Carlo Turing Machine M that decides a language L is a nondeterministic Turing Machine with exactly two choices in each step and the following conditions:

- if $x \in L$ then at least half of the computations on x halt with *yes*
 - if $x \notin L$ then all computations halt with *no*
- randomized algorithms can be modeled with an ordinary non-deterministic Turing Machine with different interpretation of the meaning of accepting the input
 - no false positive answers+
 - probability of false negatives is at most $\frac{1}{2}$

RP

Complexity class of all languages that are decided with polynomial Monte Carlo Turing Machines is denoted as RP.

- RP lies between P and NP ($P \subseteq RP \subseteq NP$)

TODO

More detail on definition of Monte Carlo Turing Machine.

Probability explanation

More details on RP

Questions:

More a to do than a question: More details on definition of Monte Carlo Turing Machine.

ZPP

- Is RP closed under complement?
- RP: no false positive answers but false negative answers possible
- coRP: no false negative answers but false positive answers possible
- $RP \cap coRP$ seems interesting:
 - problem in this class has two Monte Carlo algorithms:
 - * one has no false positives
 - * one has no false negatives
 - run enough experiments on both: right answer will come up
(either a positive answer from the one with no false positives or a negative answer from the one with no false negatives)
 - correct answer will be known for sure
 - execute both algorithms independent k times: probability that the correct answer is not obtained is $\frac{1}{2^k}$

LasVegas algorithm

The LasVegas algorithm as a Monte Carlo algorithm and its complement (one has no false positives and one has no false negatives). It runs k independent experiments on both and the right answer will come up. (Either a positive answer from the one with no false positives or a negative answer from the one with no false negatives.)

ZPP

$RP \cap coRP$

Complexity class of all languages that are decided with LasVegas algorithms is denoted as RP.

•

TODO

Questions:

PP

PP

content

•

TODO

Questions:

NP \subseteq PP

-

TODO

Questions:

BPP

BPP

Answers (*yes* and *no*) are correct with probability $\frac{3}{4}$.

-

TODO

Questions:

Randomized complexity classes

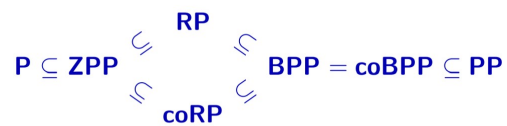


Figure 4: Relations between classes from the lecture slides [1]

RP \subseteq NP \subseteq PP

Figure 5: Relations between classes from the lecture slides [1]

TODO

Questions:

- some slides about stuff that is not in the book
- Satz von Rice

TODO

some slides about stuff that is not in the book

Satz von Rice

Questions:

References

- [1] Martin Berglund. *Lecture notes in Computational Complexity*.
- [2] *Bipartite graph image source*. https://en.wikipedia.org/wiki/Bipartite_graph.
- [3] GeeksforGeeks. *Maximum Bipartite Matching*. <https://www.geeksforgeeks.org/maximum-bipartite-matching/>, last opened: 09.12.22.
- [4] Swastik Kopparty. *Bipartite Graphs and Matchings*. <https://sites.math.rutgers.edu/~sk1233/courses/graphtheory-F11/matching.pdf>, last opened 09.12.22. 2011.
- [5] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1994.