Review & Plan

Review & Plan
Today’s plan

- Online bipartite matching (marriage problem)
- Review of the exam (how it will look)
- Concluding remarks
Online Bipartite Matching (Marriage problem)
Online Bipartite Matching

Given a bipartite graph, create a matching of maximum size

In the online setting, vertices in one side of the graph are given, and vertices in the other side arrive one by one. Upon arrival of vertex $x$, all edges connecting $x$ to its neighbors on the left are revealed. An online algorithm should match $x$ with another vertex, if possible, without any information about future vertices.

Greedy algorithm: match with any vertex on the right if possible!
Online Bipartite Matching

- Given a bipartite graph, create a matching of maximum size
- In the online setting, vertices in one side of the graph are given, and vertices in the other side arrive one by one
  - Upon arrival of vertex $x$, all edges connecting $x$ to its neighbors on the left are revealed
  - An online algorithm should match $x$ with another vertex, if possible, without any information about future vertices

**Greedy algorithm:** match with any vertex on the right if possible!
Online Bipartite Matching

- Given a bipartite graph, create a matching of maximum size
- In the online setting, vertices in one side of the graph are given, and vertices in the other side arrive one by one
  - Upon arrival of vertex $x$, all edges connecting $x$ to its neighbors on the left are revealed
  - An online algorithm should match $x$ with another vertex, if possible, without any information about future vertices
- **Greedy algorithm:** match with any vertex on the right if possible!
Online Matching

Online Bipartite Matching

- Given a bipartite graph, create a matching of maximum size
- In the online setting, vertices in one side of the graph are given, and vertices in the other side arrive one by one
  - Upon arrival of vertex \( x \), all edges connecting \( x \) to its neighbors on the left are revealed
  - An online algorithm should match \( x \) with another vertex, if possible, without any information about future vertices

**Greedy algorithm**: match with any vertex on the right if possible!
Online Matching

Online Bipartite Matching

- Given a bipartite graph, create a matching of maximum size
- In the online setting, vertices in one side of the graph are given, and vertices in the other side arrive one by one
  - Upon arrival of vertex $x$, all edges connecting $x$ to its neighbors on the left are revealed
  - An online algorithm should match $x$ with another vertex, if possible, without any information about future vertices

**Greedy algorithm**: match with any vertex on the right if possible!
Online Matching

Online Bipartite Matching

- Given a bipartite graph, create a matching of maximum size
- In the online setting, vertices in one side of the graph are given, and vertices in the other side arrive one by one
  - Upon arrival of vertex $x$, all edges connecting $x$ to its neighbors on the left are revealed
  - An online algorithm should match $x$ with another vertex, if possible, without any information about future vertices
- **Greedy algorithm:** match with any vertex on the right if possible!
Online Matching

Online Bipartite Matching

- Given a bipartite graph, create a matching of maximum size
- In the online setting, vertices in one side of the graph are given, and vertices in the other side arrive one by one
  - Upon arrival of vertex $x$, all edges connecting $x$ to its neighbors on the left are revealed
  - an online algorithm should match $x$ with another vertex, if possible, without any information about future vertices

**Greedy algorithm:** match with any vertex on the right if possible!
Online Bipartite Matching

- Given a bipartite graph, create a matching of maximum size
- In the online setting, vertices in one side of the graph are given, and vertices in the other side arrive one by one
  - Upon arrival of vertex \( x \), all edges connecting \( x \) to its neighbors on the left are revealed
  - an online algorithm should match \( x \) with another vertex, if possible, without any information about future vertices

**Greedy algorithm:** match with any vertex on the right if possible!
Online Matching

**Competitive ratio for max. problems**

- In this example, $\text{OPT}$ has a **benefit** of 4 (a perfect matching) while greedy has a benefit of 3!

- Matching is a maximization problem: we would like to maximize the benefit instead of minimizing the cost

- Competitive ratio is often defined as the maximum value of
  $$\frac{\text{Benefit(OPT)}}{\text{benefit(Alg)}}$$

**Theorem**

*Greedy has a competitive ratio of at most 2.*
Greedy Bipartite Matching

- Greedy algorithm always creates a maximal matching
  - All neighbors of an unmatched vertex $u$ are matched (otherwise greedy would have matched $u$ with one of its neighbors).
Greedy algorithm always creates a maximal matching.

- All neighbors of an unmatched vertex \( u \) are matched (otherwise greedy would have matched \( u \) with one of its neighbors).

Consider a bipartite graph with \( n \) vertices on the left.

If \( X \) denote the number of vertices on left which are matched by \( \text{OPT} \) and unmatched by greedy, all neighbors of these vertices are matched by greedy:

- Greedy matches \( n - X \) vertices on the left
- Greedy matches at least \( X \) vertex on the right

Size of greedy matching is at least \( \max \{ n - X, X \} \) and size of \( \text{OPT} \) matching is at most \( n \)

Competitive ratio will be \( \frac{n}{\max \{ n - X, X \}} \leq \frac{n}{n/2} = 2 \)
Greedy Bipartite Matching

Theorem

*Greedy has a competitive ratio of at most 2.*

- We proved an upper bound before (i.e., we showed c.r. $\leq 2$).
- For the lower bound, consider the following example:
  - Greedy matches $n/2$ vertices while $\text{OPT}$ matches $n$ vertices.
Theorem

Greedy has a competitive ratio of at most 2.

- We proved an upper bound before (i.e., we showed c.r. $\leq 2$).
- For the lower bound, consider the following example:
  - Greedy matches $n/2$ vertices while $\text{OPT}$ matches $n$ vertices.
Greedy Bipartite Matching

Theorem

*Greedy has a competitive ratio of at most 2.*

- We proved an upper bound before (i.e., we showed $c.r. \leq 2$).
- For the lower bound, consider the following example:
  - Greedy matches $n/2$ vertices while $OPT$ matches $n$ vertices
Greedy Bipartite Matching

Theorem

Greedy has a competitive ratio of at most 2.

- We proved an upper bound before (i.e., we showed $c.r. \leq 2$).
- For the lower bound, consider the following example:
  - Greedy matches $n/2$ vertices while $\text{OPT}$ matches $n$ vertices

![Diagram showing Greedy matching half of vertices compared to OPT matching all vertices](image_url)
Greedy Bipartite Matching

Theorem

*Greedy has a competitive ratio of at most 2.*

- We proved an upper bound before (i.e., we showed c.r. \( \leq 2 \)).
- For the lower bound, consider the following example:
  - Greedy matches \( n/2 \) vertices while OPT matches \( n \) vertices

![Diagram showing a competitive ratio example](image-url)
**Theorem**

Greedy has a competitive ratio of at most 2.

- We proved an upper bound before (i.e., we showed \( \text{c.r.} \leq 2 \)).
- For the lower bound, consider the following example:
  - Greedy matches \( \frac{n}{2} \) vertices while \( \text{OPT} \) matches \( n \) vertices
Online Matching

Greedy Bipartite Matching

Theorem

Greedy has a competitive ratio of at most 2.

- We proved an upper bound before (i.e., we showed c.r. $\leq 2$).
- For the lower bound, consider the following example:
  - Greedy matches $n/2$ vertices while $\text{OPT}$ matches $n$ vertices

![Graph Example]

COMP 7720 - Online Algorithms  Online Graph Problems & Goodbye Notes
Greedy Bipartite Matching

Theorem

Greedy has a competitive ratio of at most 2.

We proved an upper bound before (i.e., we showed c.r. \( \leq 2 \)).

For the lower bound, consider the following example:

- Greedy matches \( n/2 \) vertices while \( \text{OPT} \) matches \( n \) vertices.
Greedy Bipartite Matching

Theorem

Greedy has a competitive ratio of at most 2.

- We proved an upper bound before (i.e., we showed c.r. \( \leq 2 \)).
- For the lower bound, consider the following example:
  - Greedy matches \( \frac{n}{2} \) vertices while OPT matches \( n \) vertices.
Theorem

Greedy has a competitive ratio of at most 2.

- We proved an upper bound before (i.e., we showed c.r. \( \leq 2 \)).
- For the lower bound, consider the following example:
  - Greedy matches \( n/2 \) vertices while Opt matches \( n \) vertices
Online Matching

Greedy Bipartite Matching

Theorem

*Greedy has a competitive ratio of at most 2.*

We proved an upper bound before (i.e., we showed c.r. $\leq 2$).

For the lower bound, consider the following example:

- Greedy matches $n/2$ vertices while $\text{OPT}$ matches $n$ vertices
Online Matching

Greedy Bipartite Matching

**Theorem**

*Greedy has a competitive ratio of at most 2.*

- We proved an upper bound before (i.e., we showed c.r. $\leq 2$).
- For the lower bound, consider the following example:
  - Greedy matches $n/2$ vertices while OPT matches $n$ vertices
Deterministic Bipartite Matching

**Theorem**

*No deterministic online algorithm has a competitive ratio better than 2.*

- Proof similar to the case of Greedy.
- So, Greedy is the best deterministic algorithm.
- A randomized algorithm which chooses random match has also a competitive ratio of 2 (is it good?).
Rank Algorithm:

- Initially, choose a random permutation of all vertices on left
- Upon arrival of a vertex $u$ on right:
  - Let $N(u)$ be the set of unmatched neighbors of $u$
  - If $N(u) \neq 0$, match $u$ to the vertex $v \in N(u)$ with minimum index in the permutation
Online Matching

Deterministic Bipartite Matching

Rank Algorithm:

- Initially, choose a random permutation of all vertices on left
- Upon arrival of a vertex \( u \) on right:
  - Let \( N(u) \) be the set of unmatched neighbors of \( u \)
  - If \( N(u) \neq 0 \), match \( u \) to the vertex \( v \in N(u) \) with minimum index in the permutation

```
3
5
1
6
4
2
```
Online Matching

Deterministic Bipartite Matching

Rank Algorithm:

- Initially, choose a random permutation of all vertices on left
- Upon arrival of a vertex $u$ on right:
  - Let $N(u)$ be the set of unmatched neighbors of $u$
  - If $N(u) \neq 0$, match $u$ to the vertex $v \in N(u)$ with minimum index in the permutation
Deterministic Bipartite Matching

Rank Algorithm:

- Initially, choose a random permutation of all vertices on left
- Upon arrival of a vertex $u$ on right:
  - Let $N(u)$ be the set of unmatched neighbors of $u$
  - If $N(u) \neq 0$, match $u$ to the vertex $v \in N(u)$ with minimum index in the permutation
Deterministic Bipartite Matching

**Rank Algorithm:**

- Initially, choose a random permutation of all vertices on left
- Upon arrival of a vertex $u$ on right:
  - Let $N(u)$ be the set of unmatched neighbors of $u$
  - If $N(u) \neq 0$, match $u$ to the vertex $v \in N(u)$ with minimum index in the permutation

![Graph Diagram](image)
**Online Matching**

**Deterministic Bipartite Matching**

**Rank Algorithm:**

- Initially, choose a random permutation of all vertices on left
- Upon arrival of a vertex $u$ on right:
  - Let $N(u)$ be the set of unmatched neighbors of $u$
  - If $N(u) \neq 0$, match $u$ to the vertex $v \in N(u)$ with minimum index in the permutation

![Diagram](image.png)
Online Matching

Deterministic Bipartite Matching

Rank Algorithm:

- Initially, choose a random permutation of all vertices on left
- Upon arrival of a vertex $u$ on right:
  - Let $N(u)$ be the set of unmatched neighbors of $u$
  - If $N(u) \neq 0$, match $u$ to the vertex $v \in N(u)$ with minimum index in the permutation

![Diagram](image-url)

\[8 / 9\]
**Online Matching**

**Deterministic Bipartite Matching**

**Rank Algorithm:**

- Initially, choose a random permutation of all vertices on left
- Upon arrival of a vertex $u$ on right:
  - Let $N(u)$ be the set of unmatched neighbors of $u$
  - If $N(u) \neq 0$, match $u$ to the vertex $v \in N(u)$ with minimum index in the permutation
Online Matching

Deterministic Bipartite Matching

Rank Algorithm:

- Initially, choose a random permutation of all vertices on left
- Upon arrival of a vertex $u$ on right:
  - Let $N(u)$ be the set of unmatched neighbors of $u$
  - If $N(u) \neq 0$, match $u$ to the vertex $v \in N(u)$ with minimum index in the permutation
**Deterministic Bipartite Matching**

**Rank Algorithm:**

- Initially, choose a random permutation of all vertices on left
- Upon arrival of a vertex $u$ on right:
  - Let $N(u)$ be the set of unmatched neighbors of $u$
  - If $N(u) \neq 0$, match $u$ to the vertex $v \in N(u)$ with minimum index in the permutation

```
3 5
5 1
1 6
6 4
4 2
```
Rank Algorithm:

- Initially, choose a random permutation of all vertices on left
- Upon arrival of a vertex $u$ on right:
  - Let $N(u)$ be the set of unmatched neighbors of $u$
  - If $N(u) \neq 0$, match $u$ to the vertex $v \in N(u)$ with minimum index in the permutation

**Theorem**

*Rank has a competitive ratio of* $\frac{e}{e-1} \approx 1.58$
Greedy algorithm has a competitive ratio of 2 and it is the best that a deterministic algorithm can achieve.

Rank is a simple randomized algorithm with competitive ratio of $e/(e - 1)$.

It is known that no randomized, online algorithm can achieve a competitive ratio better than $e/(e - 1)$. 

Online Bipartite Matching Summary
Online Bipartite Matching Summary

- Greedy algorithm has a competitive ratio of 2 and it is the best that a deterministic algorithm can achieve.
- Rank is a simple randomized algorithm with competitive ratio of $\frac{e}{e - 1}$.
- It is known that no randomized, online algorithm can achieve a competitive ratio better than $\frac{e}{e - 1}$. 