Stable Matching in the Housing Market

Ian Alberto Paulino Vélez

December 13, 2022

Algorithms and Data Structure

Agenda

01 Background

02 Preference-ranking algorithm

03 Gale-Shapley algorithm

04 Discussion and Limitations

01 Background

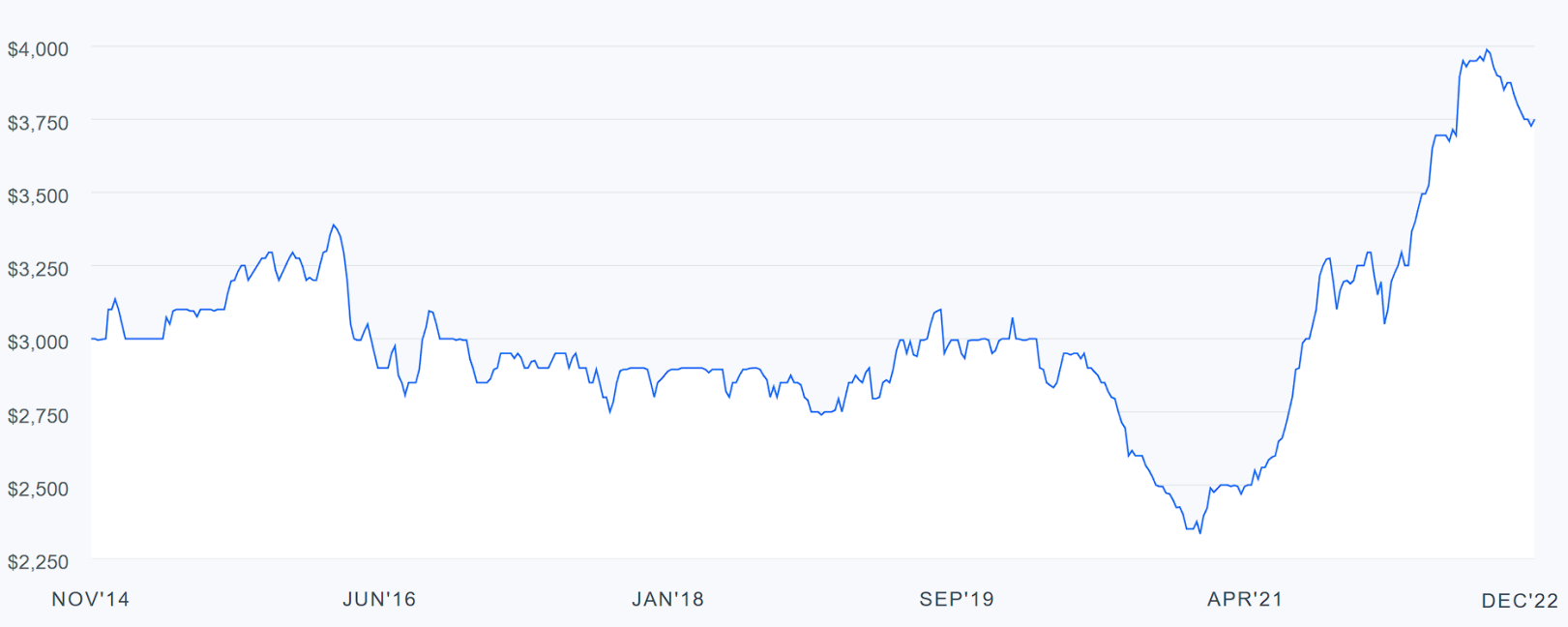
The New York City rental housing market is becoming increasingly pricier, tighter and more demanding.

→

Unbalanced power dynamics that put potential tenants at a disadvantage.

→

Prompts the need for a better rental system that reduces cost, time, and tenant-landlord conflict.



(2022, December 11). Average Rent in New York, NY and Cost Information. Zumper. https://www.zumper.com/rent-research/new-york-ny

Data Science Applications on the Housing Market

- Generating housing price indices
- Automated Valuation Models
- Matching Tenants and Landlords
- Time series forecasting
- Geographic Information Systems

Stable Matching Problem

↓

Ordering of all other subjects for each subject by preference

↓

Gale-Shapley Algorithm

02 Preference-ranking algorithm

```
Input: tenant_list, landlord_list
Output: tenant_preferred_landlords_sorted,
landlord_preferred_set_sorted

for t in tenant_list:
  for l in landlord_list:
    score = 0
    if tenant.preferences[sq.feet] <= landlord.sq_feet:
      score += 1
    if tenant.preferences[rent] >= landlord.rent:
      score += 1
    for i=2 to len(tenant.preferences):
      if tenant.preferences[i] == landlord.features[i]:
        score += 1
    tenant.ranking[landlord] = score
sort(tenant.ranking)by value
```

```
for l in landlord_list:
  for t in tenant_list:
    score = 0
    if landlord.preferences[income] <=
tenant.income and
landlord.preferences[credit_score] <=
tenant.credit_score:
      score += 2
    else if tenant.guarantor == True:
      score += 2
    for i=2 to len(landlord.preferences):
      if landlord.preferences[i] ==
tenant.features[i]:
        score += 1
    landlord.ranking[tenant] = score
sort(tenant.ranking)by value
```

02 Preference-ranking algorithm: tenants

```
Input: tenant_list, landlord_list
Output: tenant_preferred_landlords_sorted,
landlord_preferred_set_sorted
```

- Tenant list consists of…

A list of people: → Each with individual preferences for: → And individual features:

Ray, Erwin, Michael, Pedro, Michelle…	Sq. Feet, monthly rent, location, amount of bedrooms and bathrooms…	Income, credit score, whether they have good rental history, a guarantor, references…

02 Preference-ranking algorithm: tenants

```
for t in tenant_list:
  for l in landlord_list:
    score = 0
    if tenant.preferences[sq.feet] <= landlord.sq_feet:
      score += 1
    if tenant.preferences[rent] >= landlord.rent:
      score += 1
```

```
for i=2 to len(tenant.preferences):
      if tenant.preferences[i] ==
landlord.features[i]:
        score += 1
    tenant.ranking[landlord] = score
sort(tenant.ranking)by value
```

	A1	A2	A3	A4	A5	A6	A7	A8
Ray	7	5	3	6	3	6	6	7
Erwin	7	8	2	6	3	7	5	5
Michael	7	5	3	5	5	6	4	5
Pedro	6	5	2	7	3	4	7	6
Michelle	3	5	5	4	4	2	3	5
Carmi	4	5	3	5	5	5	6	4
Rocío	7	6	2	6	3	5	7	5
Daiana	4	4	6	3	3	6	3	7

02 Preference-ranking algorithm: tenants

```
for t in tenant_list:
  for l in landlord_list:
    score = 0
    if tenant.preferences[sq.feet] <= landlord.sq_feet:
      score += 1
    if tenant.preferences[rent] >= landlord.rent:
      score += 1
```

```
for i=2 to len(tenant.preferences):
      if tenant.preferences[i] ==
landlord.features[i]:
        score += 1
    tenant.ranking[landlord] = score
sort(tenant.ranking)by value
```

Index	Ray	Erwin	Michael	Pedro	Michelle	Carmi	Rocío	Daiana
0	A1	A2	A1	A4	A2	A7	A1	A8
1	A8	A1	A6	A7	A3	A2	A7	A3
2	A4	A6	A2	A1	A8	A4	A2	A6
3	A6	A4	A4	A8	A4	A5	A4	A1
4	A7	A7	A5	A2	A5	A6	A6	A2
5	A2	A8	A8	A6	A1	A1	A8	A4
6	A3	A5	A7	A5	A7	A8	A5	A5
7	A5	A3	A3	A3	A6	A3	A3	A7

02 Preference-ranking algorithm: landlords

Similarly…

```
Input: tenant_list, landlord_list
Output: tenant_preferred_landlords_sorted,
landlord_preferred_set_sorted
```

- landlord list consists of…

A list of apartments: → Each with individual preferences for: → And individual features:

A1, A2, A3…	Income, credit score, whether they have good rental history, a guarantor, references…	Sq. Feet, monthly rent, location, amount of bedrooms and bathrooms…

02 Preference-ranking algorithm: landlords

```
for l in landlord_list:
  for t in tenant_list:
    score = 0
    if landlord.preferences[income] <= tenant.income
        and landlord.preferences[credit_score] <=
        tenant.credit_score:
          score += 2
```

```
  else if tenant.guarantor == True:
      score += 2
    for i=2 to len(landlord.preferences):
      if landlord.preferences[i] ==
         tenant.features[i]:
          score += 1
    landlord.ranking[tenant] = score
sort(tenant.ranking)by value
```

	Ray	Erwin	Michael	Pedro	Michelle	Carmi	Rocío	Daiana
A1	2	3	3	2	5	5	2	4
A2	2	3	3	2	5	5	4	4
A3	2	3	3	2	3	5	2	4
A4	3	2	4	3	4	4	3	3
A5	3	2	4	3	4	4	3	3
A6	2	3	3	2	5	5	2	4
A7	3	2	4	3	4	4	5	3
A8	2	3	3	2	5	5	2	4

02 Preference-ranking algorithm: landlords

```
for l in landlord_list:
  for t in tenant_list:
    score = 0
    if landlord.preferences[income] <= tenant.income
        and landlord.preferences[credit_score] <=
        tenant.credit_score:
          score += 2
```

```
 else if tenant.guarantor == True:
      score += 2
    for i=2 to len(landlord.preferences):
      if landlord.preferences[i] ==
         tenant.features[i]:
          score += 1
    landlord.ranking[tenant] = score
sort(tenant.ranking)by value
```

Index	A1	A2	A3	A4	A5	A6	A7	A8
0	Michelle	Michelle	Carmi	Michael	Michael	Michelle	Rocío	Michelle
1	Carmi	Carmi	Daiana	Michelle	Michelle	Carmi	Michael	Carmi
2	Daiana	Rocío	Erwin	Carmi	Carmi	Daiana	Michelle	Daiana
3	Erwin	Daiana	Michael	Ray	Ray	Erwin	Carmi	Erwin
4	Michael	Erwin	Michelle	Pedro	Pedro	Michael	Ray	Michael
5	Ray	Michael	Ray	Rocío	Rocío	Ray	Pedro	Ray
6	Pedro	Ray	Pedro	Daiana	Daiana	Pedro	Daiana	Pedro
7	Rocío	Pedro	Rocío	Erwin	Erwin	Rocío	Erwin	Rocío

03 Gale-Shapley algorithm

Index	Ray	Erwin	Michael
0	A1	A2	A1
1	A8	A1	A6
2	A4	A6	A2
3	A6	A4	A4
4	A7	A7	A5
5	A2	A8	A8
6	A3	A5	A7
7	A5	A3	A3

Ray

Erwin

Michael

Pedro

Michelle

Carmi

Rocío

Daiana

A1

A2

A3

A4

A5

A6

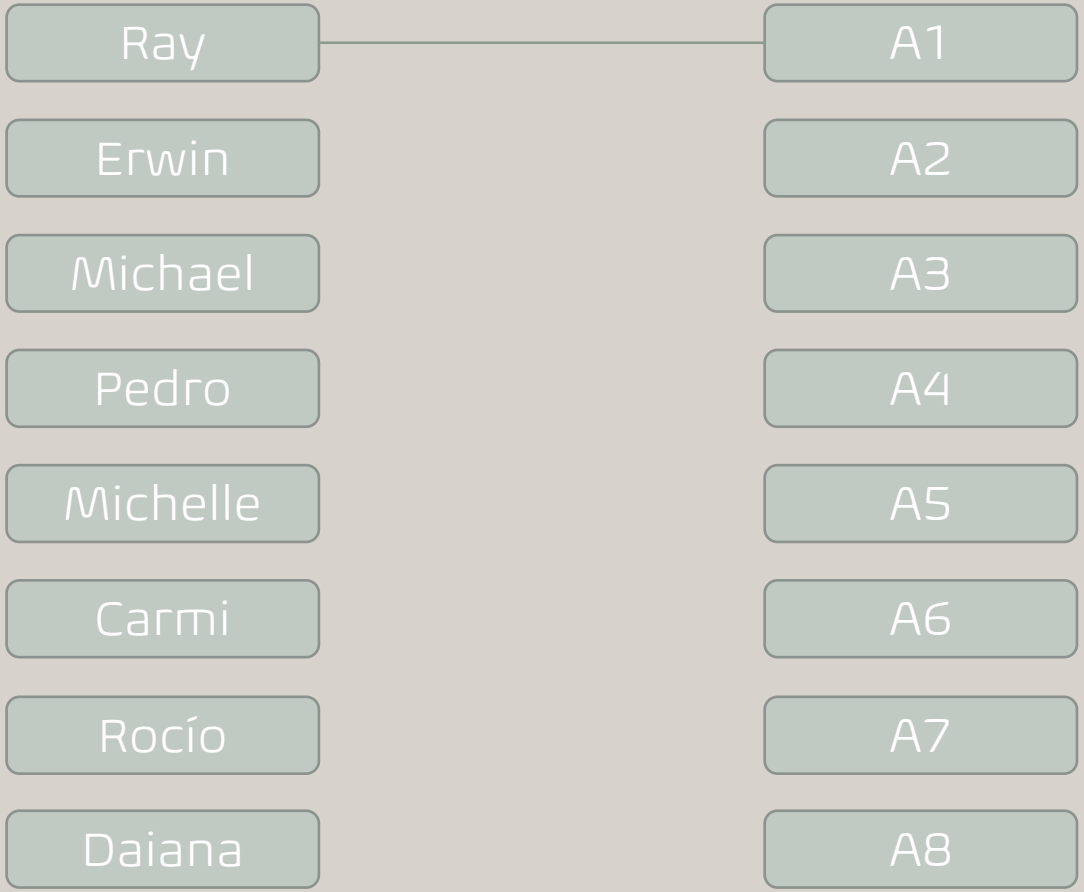
A7

A8

Index	A1
0	Michelle
1	Carmi
2	Daiana
3	Erwin
4	Michael
5	Ray
6	Pedro
7	Rocío

03 Gale-Shapley algorithm

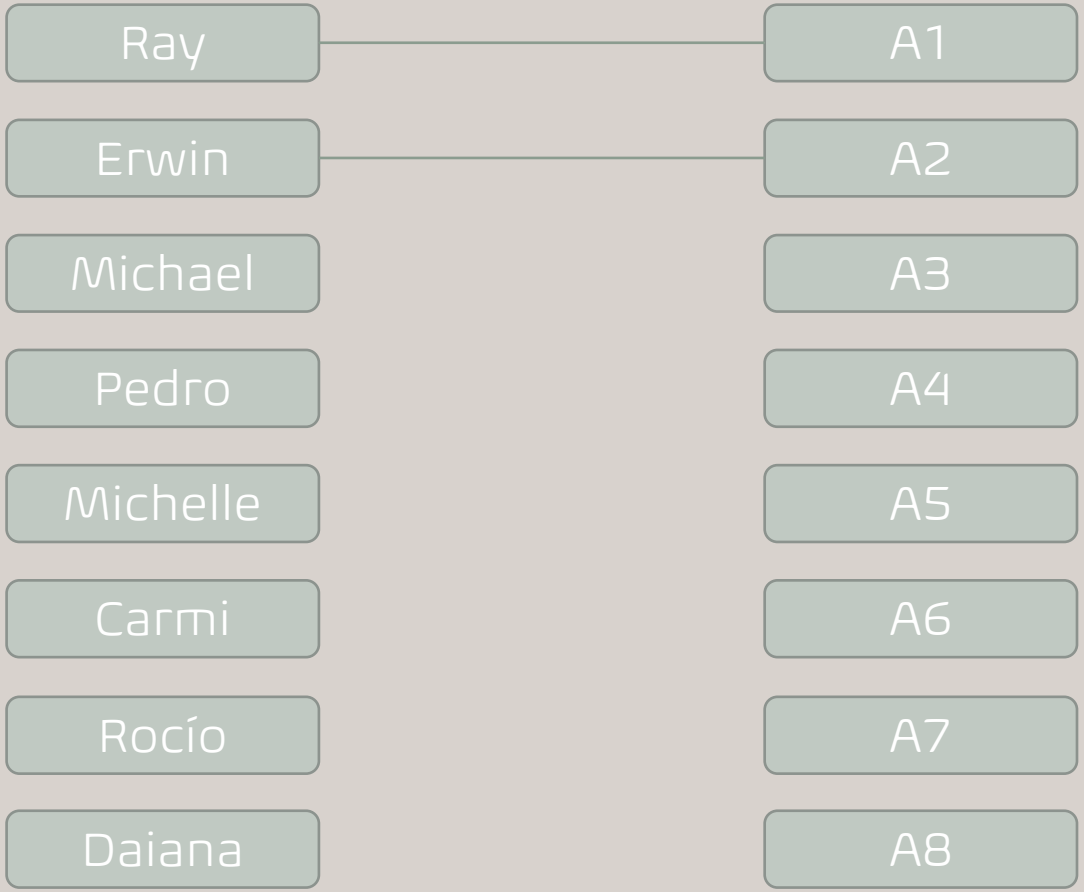
Index	Ray	Erwin	Michael
0	A1	A2	A1
1	A8	A1	A6
2	A4	A6	A2
3	A6	A4	A4
4	A7	A7	A5
5	A2	A8	A8
6	A3	A5	A7
7	A5	A3	A3



Index	A1
0	Michelle
1	Carmi
2	Daiana
3	Erwin
4	Michael
5	Ray
6	Pedro
7	Rocío

03 Gale-Shapley algorithm

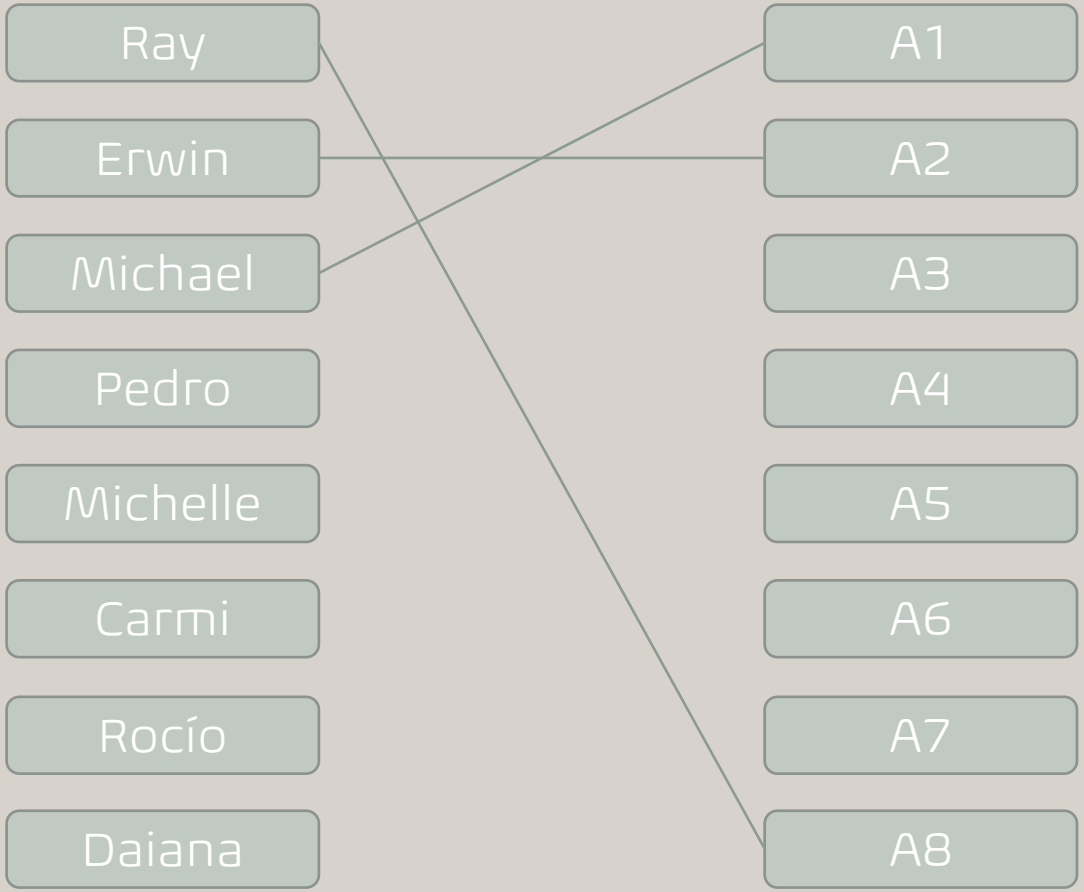
Index	Ray	Erwin	Michael
0	A1	A2	A1
1	A8	A1	A6
2	A4	A6	A2
3	A6	A4	A4
4	A7	A7	A5
5	A2	A8	A8
6	A3	A5	A7
7	A5	A3	A3



Index	A1
0	Michelle
1	Carmi
2	Daiana
3	Erwin
4	Michael
5	Ray
6	Pedro
7	Rocío

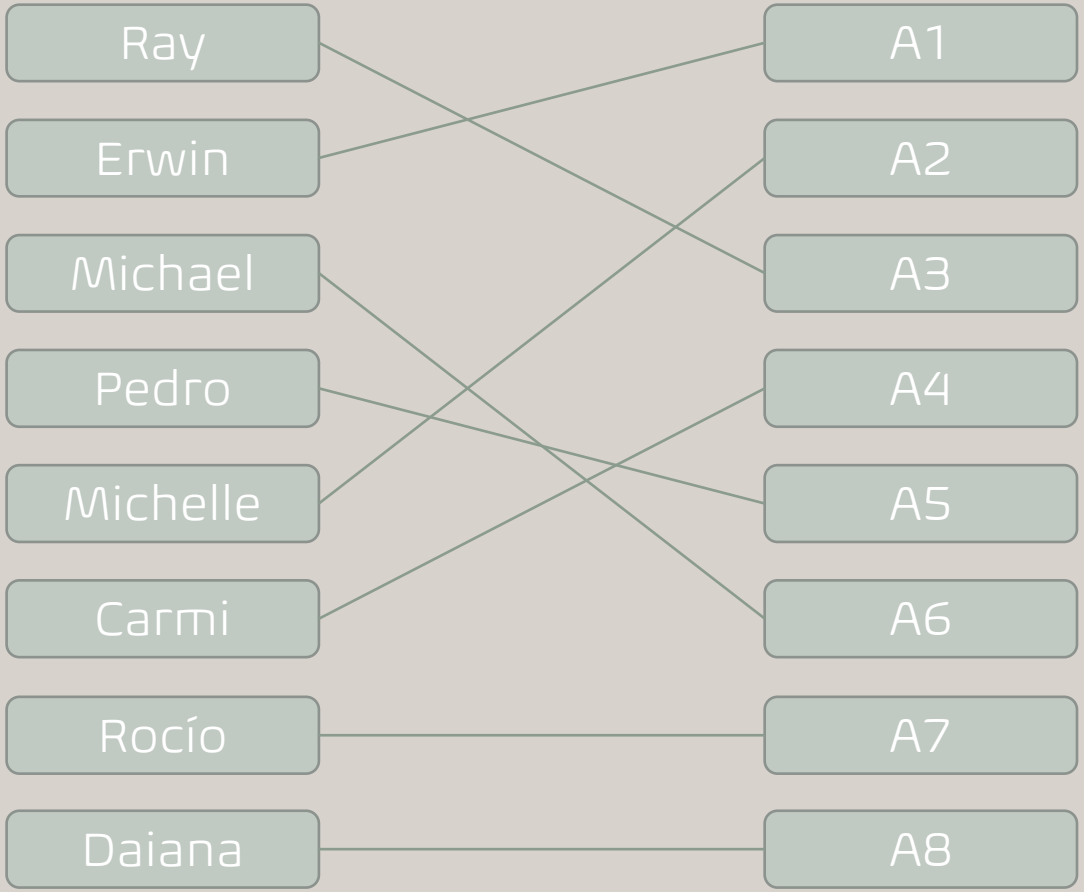
03 Gale-Shapley algorithm

Index	Ray	Erwin	Michael
0	A1	A2	A1
1	A8	A1	A6
2	A4	A6	A2
3	A6	A4	A4
4	A7	A7	A5
5	A2	A8	A8
6	A3	A5	A7
7	A5	A3	A3



Index	A1
0	Michelle
1	Carmi
2	Daiana
3	Erwin
4	Michael
5	Ray
6	Pedro
7	Rocío

03 Gale-Shapley algorithm



Discussion and limitations

Optimality

The algorithm is tenant-optimal.

Potential improvements

Better attribute specification and weighting of preferences.

Time complexity

Runs in $O(n^2)$.