Networks and Markets

Hw2 submission

Part 5: Experimental Evaluations

Alon Polski 206530461

Anna Petrenko 320460306

Ariel Chiskis 322442112

8.

(b).

The prices output by the VCG mechanism are identical to the ones output by the algorithm from Problem 7.

On the matching market frame with values as described in Lecture 5 Page 7 and on the example in q.2 the algorithms output identical prices;

```
Testing on the matching market frame with values as described in Lecture 5 Page 7:

Results for the procedure constructed in Theorem 8.2:
Matching:  M(0)=1, M(1)=0, M(2)=2
Item prices:  p(0)=0, p(1)=3, p(2)=2

Results for the VCG algorithm:
Matching:  M(0)=1, M(1)=0, M(2)=2
Item prices:  p(0)=0, p(1)=3, p(2)=2
```

```
Testing on q2.a:

Results for the procedure constructed in Theorem 8.2:
Matching:  M(0)=1, M(1)=0, M(2)=2
Item prices:  p(0)=1, p(1)=2, p(2)=0

Results for the VCG algorithm:
Matching:  M(0)=1, M(1)=0, M(2)=2
Item prices:  p(0)=1, p(1)=2, p(2)=0
```

```
Testing on q2.b:

Results for the procedure constructed in Theorem 8.2:
Matching:  M(0)=2, M(1)=0, M(2)=1
Item prices:  p(0)=3, p(1)=0, p(2)=3

Results for the VCG algorithm:
Matching:  M(0)=2, M(1)=0, M(2)=1
Item prices:  p(0)=3, p(1)=0, p(2)=3
```

Note, for q2.b the `market_eq()` routine chose a market equilibrium matching that differs from the one presented in q.2b solution, both are valid solutions as evident from the final utilites in q.2b.

More concrete examples follow.

```
Testing on a matching market context with n>m:
Valuations:  [[4, 12], [7, 10], [7, 7]]

Results for the procedure constructed in Theorem 8.2:
Matching:  M(0)=1, M(1)=0, M(2)=None
Item prices:  p(0)=7, p(1)=10

Results for the VCG algorithm:
Matching:  M(0)=1, M(1)=0, M(2)=None
Item prices:  p(0)=7, p(1)=10
```

```
Testing on a new matching market context:
Valuations:  [[3, 3, 10], [36, 4, 24], [33, 40, 47]]

Results for the procedure constructed in Theorem 8.2:
Matching:  M(0)=2, M(1)=0, M(2)=1
Item prices:  p(0)=0, p(1)=0, p(2)=7

Results for the VCG algorithm:
Matching:  M(0)=2, M(1)=0, M(2)=1
Item prices:  p(0)=0, p(1)=0, p(2)=7
```

```
Testing on a matching market context with n<m:
Valuations:  [[3, 3, 10], [33, 40, 47]]

Results for the procedure constructed in Theorem 8.2:
Matching:  M(0)=2, M(1)=1
Item prices:  p(0)=0, p(1)=0, p(2)=7

Results for the VCG algorithm:
Matching:  M(0)=2, M(1)=1
Item prices:  p(0)=0, p(1)=0, p(2)=7
```

We run both algorithms on 100 random context examples for $n = m = 3$, the algorithms output identical results in 100% of runs. Observe last two runs and overall comparison;

```
Testing on a random context for n=3, m=3:
Valuations:  [[27, 12, 1], [37, 37, 27], [19, 38, 7]]

Results for the procedure constructed in Theorem 8.2:
Matching:  M(0)=0, M(1)=2, M(2)=1
Item prices:  p(0)=10, p(1)=10, p(2)=0

Results for the VCG algorithm:
Matching:  M(0)=0, M(1)=2, M(2)=1
Item prices:  p(0)=10, p(1)=10, p(2)=0
```

```
Testing on a random context for n=3, m=3:
Valuations:  [[22, 10, 19], [6, 36, 50], [37, 9, 26]]

Results for the procedure constructed in Theorem 8.2:
Matching:  M(0)=1, M(1)=2, M(2)=0
Item prices:  p(0)=12, p(1)=0, p(2)=9

Results for the VCG algorithm:
Matching:  M(0)=1, M(1)=2, M(2)=0
Item prices:  p(0)=12, p(1)=0, p(2)=9
----------------------------------------------------------------------
Identical results in 100 out of 100 runs
```

We run both algorithms on 100 random context examples for $n = m = 20$, the algorithms output identical results in 100% of runs. Observe last two runs and overall comparison;

```
Testing on a random context for n=20, m=20:
Valuations:  [[25, 9, 3, 39, 18, 12, 1, 49, 12, 25, 30, 8, 28, 31, 31, 5, 22, 38, 8, 49], [27, 2, 36, 50, 17, 33, 34, 15, 38, 40, 38, 19, 25, 5, 35, 31, 5, 47, 26,
 24], [43, 19, 6, 43, 30, 21, 41, 23, 49, 43, 12, 10, 31, 34, 25, 39, 37, 18, 27, 11], [34, 32, 20, 26, 28, 40, 14, 16, 19, 7, 49, 31, 31, 30, 21, 14, 23, 7, 30,
 36], [12, 28, 47, 48, 15, 26, 38, 37, 45, 10, 22, 32, 41, 32, 13, 41, 26, 25, 29, 33], [37, 7, 33, 37, 47, 11, 44, 28, 43, 18, 27, 42, 6, 14, 19, 14, 9, 16, 18,
 27], [10, 38, 41, 38, 41, 11, 14, 36, 7, 6, 50, 32, 5, 11, 19, 12, 22, 7, 50, 31], [8, 41, 3, 17, 19, 29, 21, 38, 50, 39, 49, 8, 32, 34, 35, 3, 13, 19, 16, 44],
 [14, 12, 46, 49, 36, 31, 1, 28, 1, 16, 14, 29, 29, 41, 42, 36, 28, 3, 38, 32], [17, 45, 41, 1, 41, 36, 20, 31, 11, 33, 33, 50, 12, 16, 9, 34, 45, 24, 30, 20], [15,
 34, 16, 12, 7, 27, 27, 24, 8, 16, 26, 49, 16, 17, 45, 35, 24, 19, 36, 45], [16, 49, 37, 4, 1, 37, 25, 43, 19, 21, 26, 15, 40, 24, 1, 6, 31, 11, 50, 42], [23, 36,
 12, 19, 22, 28, 5, 20, 16, 38, 34, 11, 11, 36, 20, 32, 31, 36, 18, 7], [46, 39, 20, 18, 11, 50, 34, 48, 29, 17, 43, 29, 10, 35, 19, 24, 8, 17, 21, 27], [19, 10, 16,
  1, 40, 36, 35, 2, 20, 15, 25, 15, 7, 13, 17, 19, 16, 2, 36, 22], [29, 4, 11, 24, 28, 32, 10, 23, 41, 38, 46, 31, 28, 23, 25, 30, 39, 23, 12, 21], [14, 10, 18, 26,
 16, 6, 27, 2, 32, 33, 41, 7, 15, 17, 12, 17, 27, 1, 29, 20], [30, 37, 20, 14, 27, 4, 15, 30, 27, 29, 27, 26, 30, 5, 34, 16, 8, 25, 30, 4], [27, 39, 37, 33, 43, 50,
 20, 37, 27, 28, 18, 18, 1, 2, 40, 30, 42, 38, 49, 46], [19, 7, 7, 10, 4, 37, 27, 37, 47, 40, 32, 2, 49, 33, 5, 31, 38, 17, 37, 24]]

Results for the procedure constructed in Theorem 8.2:
Matching:  M(0)=7, M(1)=17, M(2)=15, M(3)=10, M(4)=2, M(5)=6, M(6)=18, M(7)=8, M(8)=3, M(9)=11, M(10)=19, M(11)=1, M(12)=13, M(13)=0, M(14)=4, M(15)=16, M(16)=9,
 M(17)=14, M(18)=5, M(19)=12
Item prices:  p(0)=4, p(1)=9, p(2)=1, p(3)=4, p(4)=5, p(5)=8, p(6)=2, p(7)=6, p(8)=10, p(9)=4, p(10)=12, p(11)=10, p(12)=2, p(13)=0, p(14)=6, p(15)=0, p(16)=5,
 p(17)=0, p(18)=10, p(19)=6

Results for the VCG algorithm:
Matching:  M(0)=7, M(1)=17, M(2)=15, M(3)=10, M(4)=2, M(5)=6, M(6)=18, M(7)=8, M(8)=3, M(9)=11, M(10)=19, M(11)=1, M(12)=13, M(13)=0, M(14)=4, M(15)=16, M(16)=9,
 M(17)=14, M(18)=5, M(19)=12
Item prices:  p(0)=4, p(1)=9, p(2)=1, p(3)=4, p(4)=5, p(5)=8, p(6)=2, p(7)=6, p(8)=10, p(9)=4, p(10)=12, p(11)=10, p(12)=2, p(13)=0, p(14)=6, p(15)=0, p(16)=5,
 p(17)=0, p(18)=10, p(19)=6
```

```
Testing on a random context for n=20, m=20:
Valuations:  [[21, 5, 11, 24, 38, 39, 19, 31, 1, 19, 7, 15, 34, 6, 17, 8, 41, 43, 16, 10], [14, 1, 14, 4, 15, 46, 46, 1, 22, 34, 47, 24, 19, 17, 25, 42, 16, 34, 50,
 5], [1, 12, 17, 28, 13, 13, 25, 1, 18, 38, 25, 7, 22, 2, 24, 5, 32, 5, 34, 48], [29, 26, 34, 45, 26, 28, 1, 10, 31, 4, 34, 9, 39, 41, 17, 47, 11, 25, 21, 19], [38,
 35, 22, 10, 5, 33, 1, 3, 49, 12, 32, 10, 43, 27, 20, 23, 29, 23, 43, 42], [23, 14, 46, 24, 21, 5, 21, 45, 28, 36, 2, 45, 41, 26, 38, 14, 25, 12, 30, 37], [12, 31,
 49, 26, 30, 8, 38, 42, 33, 48, 28, 19, 33, 41, 45, 49, 5, 4, 30, 29], [35, 47, 19, 30, 32, 14, 33, 7, 17, 33, 8, 11, 47, 39, 26, 35, 27, 2, 7, 20], [16, 24, 18, 13,
  49, 11, 21, 38, 26, 34, 40, 37, 29, 5, 41, 18, 48, 3, 35, 23], [40, 34, 44, 31, 47, 32, 39, 28, 19, 43, 14, 25, 7, 22, 4, 33, 27, 38, 17, 38], [42, 32, 49, 2, 41,
 40, 48, 12, 27, 14, 45, 12, 8, 38, 36, 26, 5, 36, 49, 41], [5, 16, 27, 33, 18, 41, 46, 42, 7, 5, 4, 11, 6, 19, 4, 23, 6, 30, 7, 38], [24, 11, 31, 19, 27, 12, 38, 6,
  11, 47, 27, 31, 44, 20, 33, 9, 15, 32, 2, 26], [1, 35, 24, 16, 7, 5, 37, 33, 41, 14, 13, 21, 5, 26, 4, 33, 47, 12, 33, 49], [29, 37, 10, 41, 12, 23, 48, 34, 2, 9,
 16, 13, 22, 15, 34, 27, 11, 27, 27, 39], [28, 12, 50, 33, 15, 43, 19, 7, 19, 24, 41, 20, 11, 12, 27, 21, 18, 16, 36, 37], [40, 49, 33, 50, 7, 26, 24, 2, 18, 16, 19,
  7, 44, 8, 50, 9, 15, 20, 41, 11], [16, 7, 43, 24, 32, 22, 2, 3, 8, 29, 8, 46, 17, 5, 12, 10, 22, 21, 14, 50], [38, 3, 7, 16, 47, 15, 34, 34, 30, 45, 50, 23, 7, 39,
 31, 12, 50, 50, 43, 36], [6, 2, 17, 18, 3, 44, 15, 21, 9, 5, 26, 15, 33, 3, 9, 5, 25, 44, 26, 11]]

Results for the procedure constructed in Theorem 8.2:
Matching:  M(0)=17, M(1)=18, M(2)=19, M(3)=13, M(4)=8, M(5)=7, M(6)=15, M(7)=1, M(8)=4, M(9)=9, M(10)=0, M(11)=6, M(12)=12, M(13)=16, M(14)=3, M(15)=2, M(16)=14,
 M(17)=11, M(18)=10, M(19)=5
Item prices:  p(0)=3, p(1)=1, p(2)=10, p(3)=4, p(4)=10, p(5)=6, p(6)=11, p(7)=7, p(8)=3, p(9)=6, p(10)=7, p(11)=7, p(12)=3, p(13)=0, p(14)=2, p(15)=6, p(16)=9,
 p(17)=7, p(18)=10, p(19)=11

Results for the VCG algorithm:
Matching:  M(0)=17, M(1)=18, M(2)=19, M(3)=13, M(4)=8, M(5)=7, M(6)=15, M(7)=1, M(8)=4, M(9)=9, M(10)=0, M(11)=6, M(12)=12, M(13)=16, M(14)=3, M(15)=2, M(16)=14,
 M(17)=11, M(18)=10, M(19)=5
Item prices:  p(0)=3, p(1)=1, p(2)=10, p(3)=4, p(4)=10, p(5)=6, p(6)=11, p(7)=7, p(8)=3, p(9)=6, p(10)=7, p(11)=7, p(12)=3, p(13)=0, p(14)=2, p(15)=6, p(16)=9,
 p(17)=7, p(18)=10, p(19)=11
----------------------------------------------------------------------
Identical results in 100 out of 100 runs
```

Bonus question 2.

(a). We implemented the bonus routine `random_bundles_valuations(n,m)` and the accompanying `run_vcg_on_random_bundles_valuations()` to run the VCG pricing algorithm on the randomly chosen context.

To simplify the analysis we also implemented the sorted version, `random_bundles_valuations_sorted(n, m)` similarly generates a matching market context for m bundles of identical goods, where each of n buyers has a random value for an individual good (between 1 to 50; ties are allowed), but the buyers are sorted in ascending order of value per good s.t the buyer with highest value per good is called buyer no. n.

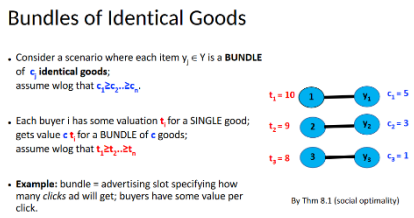
`run_vcg_on_random_bundles_valuations_sorted()` runs the VCG pricing algorithm on the randomly chosen sorted context, for $n = m = 20$.

The analysis can be applied w.l.o.g to the sorted version.

As demonstrated in simulation runs below, the results make perfect sense in the bundles context. As expected from the bundles context analysis, the buyer with the highest value per good is matched to the biggest bundle and so on;

Denoting the buyer i's valuation for an individual good by t_i and the number of goods in bundle j by $c_j = j$,
in the sorted version of random bundles context,

$$t_0 \le t_1 \le \cdots \le t_{n-1}, \quad c_0 \le c_1 \le \cdots \le c_{m-1}$$



and as in the analysis in lec.5, an allocation M^* that maximizes SV is

$$M^*(i) = i$$

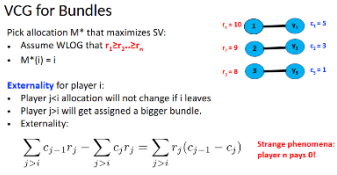
(The largest bundle must go to the person who values the good (and hence the bundle of goods) the most and so on).

This is the matching chosen by the VCG algorithm implementation, complying with the analysis in lec.7.

The player with the lowest value per good (player 0 in this context) indeed paid 0 according to our VCG implementation.

The prices returned by the VCG alg. are identical to the externality prices computed directly from the analysis in lec.7 as

$$p_i = \sum_{j<i} r_j\left(c_{j+1} - c_j\right) = \sum_{j<i} r_j(j+1-j) = \sum_{j<i} r_j$$



(note, the affected players are $j < i$ as the buyers are sorted in ascending order).

The results also comply with thm8.3;

Theorem 8.3. *For any matching market for bundles* $\Gamma = (n, Y, \vec{v})$ *defined by* $\vec{c}, \vec{t}$, *and every market equilibrium* (M, p) *for* Γ, *it holds that for every* $j = 1, \ldots, n-1$,

$$\frac{p(y_j)}{c_j} \geq \frac{p(y_{j+1})}{c_{j+1}}.$$

(in reversed order in this context)
This ratio is output in the results for bundle indices $j = 1, \ldots, n-1$.

Several results for analysis follow;

```
Bonus 2(a), random bundles valuations (sorted):
Buyers are sorted in the ascending order of their value per good.
Buyers value per good:  [4, 9, 13, 15, 17, 18, 18, 22, 23, 24, 25, 31, 31, 32, 32, 35, 37, 42, 46, 49]
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 4.00, 13.00, 26.00, 41.00, 58.00, 76.00, 94.00, 116.00, 139.00, 163.00, 188.00, 219.00, 250.00, 282.00, 314.00, 349.00, 386.00, 428.00, 474.00]
Externality prices:
[0.00, 4.00, 13.00, 26.00, 41.00, 58.00, 76.00, 94.00, 116.00, 139.00, 163.00, 188.00, 219.00, 250.00, 282.00, 314.00, 349.00, 386.00, 428.00, 474.00]
Price of bundle divided by the number of items in the bundle:
[4.00, 6.50, 8.67, 10.25, 11.60, 12.67, 13.43, 14.50, 15.44, 16.30, 17.09, 18.25, 19.23, 20.14, 20.93, 21.81, 22.71, 23.78, 24.95]
```

```
Bonus 2(a), random bundles valuations (sorted):
Buyers are sorted in the ascending order of their value per good.
Buyers value per good:  [3, 8, 12, 15, 16, 22, 23, 30, 30, 32, 33, 35, 36, 38, 40, 46, 47, 47, 49, 50]
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 3.00, 11.00, 23.00, 38.00, 54.00, 76.00, 99.00, 129.00, 159.00, 191.00, 224.00, 259.00, 295.00, 333.00, 373.00, 419.00, 466.00, 513.00, 562.00]
Externality prices:
[0.00, 3.00, 11.00, 23.00, 38.00, 54.00, 76.00, 99.00, 129.00, 159.00, 191.00, 224.00, 259.00, 295.00, 333.00, 373.00, 419.00, 466.00, 513.00, 562.00]
Price of bundle divided by the number of items in the bundle:
[3.00, 5.50, 7.67, 9.50, 10.80, 12.67, 14.14, 16.12, 17.67, 19.10, 20.36, 21.58, 22.69, 23.79, 24.87, 26.19, 27.41, 28.50, 29.58]
```

```
Bonus 2(a), random bundles valuations (sorted):
Buyers are sorted in the ascending order of their value per good.
Buyers value per good:  [1, 6, 15, 15, 17, 17, 18, 20, 20, 26, 27, 28, 30, 31, 32, 35, 38, 43, 45, 47]
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 1.00, 7.00, 22.00, 37.00, 54.00, 71.00, 89.00, 109.00, 129.00, 155.00, 182.00, 210.00, 240.00, 271.00, 303.00, 338.00, 376.00, 419.00, 464.00]
Externality prices:
[0.00, 1.00, 7.00, 22.00, 37.00, 54.00, 71.00, 89.00, 109.00, 129.00, 155.00, 182.00, 210.00, 240.00, 271.00, 303.00, 338.00, 376.00, 419.00, 464.00]
Price of bundle divided by the number of items in the bundle:
[1.00, 3.50, 7.33, 9.25, 10.80, 11.83, 12.71, 13.62, 14.33, 15.50, 16.55, 17.50, 18.46, 19.36, 20.20, 21.12, 22.12, 23.28, 24.42]
```

(b). When run in the same contexts (with the same randomness), GSP prices are consistently higher than VCG prices. Moreover, the difference increases faster than the bundle size with the GSP price for the bundle of 19 items being almost twice the price of VCG. GSP makes people pay the market equilibrium prices. As expected, for both algorithms the lowest price is always 0.

Note that VCG mechanism can DST-implement SV-max for any context, GSP mechanism is not DST, but it Nash-implements SV-max.

```
Bonus 2(b), VCG and GSP comparison for random bundles valuations (sorted):
Buyers are sorted in the ascending order of their value per good.
Buyers value per good:  [4, 7, 7, 8, 9, 9, 11, 11, 19, 19, 22, 23, 23, 27, 28, 31, 32, 36, 40, 50]
VCG:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 4.00, 11.00, 18.00, 26.00, 35.00, 44.00, 55.00, 66.00, 85.00, 104.00, 126.00, 149.00, 172.00, 199.00, 227.00, 258.00, 290.00, 326.00, 366.00]
GSP:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 4.00, 14.00, 21.00, 32.00, 45.00, 54.00, 77.00, 88.00, 171.00, 190.00, 242.00, 276.00, 299.00, 378.00, 420.00, 496.00, 544.00, 648.00, 760.00]
Item prices GSP - VCG:
[0.00, 0.00, 3.00, 3.00, 6.00, 10.00, 10.00, 22.00, 22.00, 86.00, 86.00, 116.00, 127.00, 127.00, 179.00, 193.00, 238.00, 254.00, 322.00, 394.00]
```

```
Bonus 2(b), VCG and GSP comparison for random bundles valuations (sorted):
Buyers are sorted in the ascending order of their value per good.
Buyers value per good:  [1, 3, 6, 8, 9, 12, 14, 14, 15, 15, 18, 21, 33, 35, 40, 42, 43, 45, 48, 49]
VCG:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 1.00, 4.00, 10.00, 18.00, 27.00, 39.00, 53.00, 67.00, 82.00, 97.00, 115.00, 136.00, 169.00, 204.00, 244.00, 286.00, 329.00, 374.00, 422.00]
GSP:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 1.00, 6.00, 18.00, 32.00, 45.00, 72.00, 98.00, 112.00, 135.00, 150.00, 198.00, 252.00, 429.00, 490.00, 600.00, 672.00, 731.00, 810.00, 912.00]
Item prices GSP - VCG:
[0.00, 0.00, 2.00, 8.00, 14.00, 18.00, 33.00, 45.00, 45.00, 53.00, 53.00, 83.00, 116.00, 260.00, 286.00, 356.00, 386.00, 402.00, 436.00, 490.00]
```

```
Bonus 2(b), VCG and GSP comparison for random bundles valuations (sorted):
Buyers are sorted in the ascending order of their value per good.
Buyers value per good:  [1, 4, 12, 12, 15, 17, 18, 20, 21, 24, 24, 28, 32, 33, 35, 35, 37, 43, 49, 49]
VCG:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 1.00, 5.00, 17.00, 29.00, 44.00, 61.00, 79.00, 99.00, 120.00, 144.00, 168.00, 196.00, 228.00, 261.00, 296.00, 331.00, 368.00, 411.00, 460.00]
GSP:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 1.00, 8.00, 36.00, 48.00, 75.00, 102.00, 126.00, 160.00, 189.00, 240.00, 264.00, 336.00, 416.00, 462.00, 525.00, 560.00, 629.00, 774.00, 931.00]
Item prices GSP - VCG:
[0.00, 0.00, 3.00, 19.00, 19.00, 31.00, 41.00, 47.00, 61.00, 69.00, 96.00, 96.00, 140.00, 188.00, 201.00, 229.00, 229.00, 261.00, 363.00, 471.00]
```

```
Bonus 2(b), VCG and GSP comparison for random bundles valuations (sorted):
Buyers are sorted in the ascending order of their value per good.
Buyers value per good:  [3, 5, 7, 12, 15, 16, 18, 19, 25, 25, 31, 32, 32, 36, 37, 40, 40, 40, 45, 48]
VCG:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 3.00, 8.00, 15.00, 27.00, 42.00, 58.00, 76.00, 95.00, 120.00, 145.00, 176.00, 208.00, 240.00, 276.00, 313.00, 353.00, 393.00, 433.00, 478.00]
GSP:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 3.00, 10.00, 21.00, 48.00, 75.00, 96.00, 126.00, 152.00, 225.00, 250.00, 341.00, 384.00, 416.00, 504.00, 555.00, 640.00, 680.00, 720.00, 855.00]
Item prices GSP - VCG:
[0.00, 0.00, 2.00, 6.00, 21.00, 33.00, 38.00, 50.00, 57.00, 105.00, 105.00, 165.00, 176.00, 176.00, 228.00, 242.00, 287.00, 287.00, 287.00, 377.00]
```

```
Bonus 2(b), VCG and GSP comparison for random bundles valuations (sorted):
Buyers are sorted in the ascending order of their value per good.
Buyers value per good:  [3, 7, 7, 8, 11, 12, 15, 16, 18, 19, 25, 36, 38, 39, 39, 40, 41, 43, 43, 47]
VCG:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 3.00, 10.00, 17.00, 25.00, 36.00, 48.00, 63.00, 79.00, 97.00, 116.00, 141.00, 177.00, 215.00, 254.00, 293.00, 333.00, 374.00, 417.00, 460.00]
GSP:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 3.00, 14.00, 21.00, 32.00, 55.00, 72.00, 105.00, 128.00, 162.00, 190.00, 275.00, 432.00, 494.00, 546.00, 585.00, 640.00, 697.00, 774.00, 817.00]
Item prices GSP - VCG:
[0.00, 0.00, 4.00, 4.00, 7.00, 19.00, 24.00, 42.00, 49.00, 65.00, 74.00, 134.00, 255.00, 279.00, 292.00, 292.00, 307.00, 323.00, 357.00, 357.00]
```

```
Bonus 2(b), VCG and GSP comparison for random bundles valuations (sorted):
Buyers are sorted in the ascending order of their value per good.
Buyers value per good:  [2, 4, 5, 5, 13, 14, 18, 19, 26, 27, 28, 28, 28, 32, 33, 34, 38, 40, 41, 42]
VCG:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 2.00, 6.00, 11.00, 16.00, 29.00, 43.00, 61.00, 80.00, 106.00, 133.00, 161.00, 189.00, 217.00, 249.00, 282.00, 316.00, 354.00, 394.00, 435.00]
GSP:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 2.00, 8.00, 15.00, 20.00, 65.00, 84.00, 126.00, 152.00, 234.00, 270.00, 308.00, 336.00, 364.00, 448.00, 495.00, 544.00, 646.00, 720.00, 779.00]
Item prices GSP - VCG:
[0.00, 0.00, 2.00, 4.00, 4.00, 36.00, 41.00, 65.00, 72.00, 128.00, 137.00, 147.00, 147.00, 147.00, 199.00, 213.00, 228.00, 292.00, 326.00, 344.00]
```

Interestingly, for a context where all players have identical value per good, both algorithms return identical results.

```
Bonus 2(b), VCG and GSP comparison for bundles with constant valuations:
Buyers value per good:  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
VCG:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 1.00, 2.00, 3.00, 4.00, 5.00, 6.00, 7.00, 8.00, 9.00, 10.00, 11.00, 12.00, 13.00, 14.00, 15.00, 16.00, 17.00, 18.00, 19.00]
GSP:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 1.00, 2.00, 3.00, 4.00, 5.00, 6.00, 7.00, 8.00, 9.00, 10.00, 11.00, 12.00, 13.00, 14.00, 15.00, 16.00, 17.00, 18.00, 19.00]
Item prices GSP - VCG:
[0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00]
```

```
Bonus 2(b), VCG and GSP comparison for bundles with constant valuations:
Buyers value per good:  [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100]
VCG:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 100.00, 200.00, 300.00, 400.00, 500.00, 600.00, 700.00, 800.00, 900.00, 1000.00, 1100.00, 1200.00, 1300.00, 1400.00, 1500.00, 1600.00, 1700.00, 1800.00,
 1900.00]
GSP:
Matching:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Item prices:
[0.00, 100.00, 200.00, 300.00, 400.00, 500.00, 600.00, 700.00, 800.00, 900.00, 1000.00, 1100.00, 1200.00, 1300.00, 1400.00, 1500.00, 1600.00, 1700.00, 1800.00,
 1900.00]
Item prices GSP - VCG:
[0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00]
```

9. We think the most logical thing would be to define the value on the edge between rider i and driver j as:

$$(*)\ (rider\ i's\ value) - c(i,j)$$

if that value is positive, otherwise the value will be 0.
That's because it seems that in an exchange network setting, it's reasonable to view the value on the edge between rider i and driver j as the value driver j has for rider i's ride and $(*)$ can capture that value because the driver will get $(rider's\ value)$ paid for giving that ride, however, the driver will have to travel from his location to the rider's location and then from that location to the rider's destination, which decreases his revenue by a function of the distance travelled (because it wastes fuel and time.), this also decreases rider i's value for the match due to time wastage.
The reason we added that zero evaluation when $(*)$ is negative is this discussion on the forums.

10b. The results are as follows:

- For 10 riders and 10 riders:
  Average rider profits: 16.485
  Average price: 2.131
- For 20 riders and 5 drivers:
  Average rider profits: 4.629
  Average price: 19.706
- For 5 riders and 20 drivers:
  Average rider profits: 26.284
  Average price: 0.312

(The way we calculated rider's profits is their utility in the matching markets reduction, the way we calculated price is utility of a matched drivers. Because it would be weird to consider the utility of an unmatched driver a price.)

First, we can notice that if there are much more drivers than riders, riders will get more profits than if there were the same number of riders and drivers, also in that case riders will get more profits than the drivers.
When there are more riders than drivers, riders will get even less value than if there were the same number of riders and drivers and riders will earn less than drivers. The same is true vice versa;
If there are more riders than drivers, drivers will get more value than if there is the same number of riders and drivers. When there are more drivers than riders, drivers will get even less value than if there was the same number of drivers and riders.
That seems like a very truthful representation of a property of real markets, as supply grows bigger than the demand, prices will decrease, and the consumers will have the upper hand. As supply becomes smaller than demand, prices grow

and sellers will have the upper hand.
Second interesting phenomenon we can notice is that when there is the same number of drivers and riders, riders will earn more than drivers, also, riders earned more when there were 5 riders than how much drivers earned when there were 5 drivers, and also, riders earned more when there were 20 riders than how much drivers earned when there were 20 drivers, this backs up the suggestion from the answer to question 2(c), the says that probably the market equilibrium algorithm prioritizes the buyers.

11. We can multiply a rider's value, v, by a factor, c, and replace a rider's value by cv.
c should be a function of the rider's location and destination, perhaps a sum $(x + y)$, where x is determined by the rider's location and y is determined by the rider's destination. c will reflect how popular are the rider's location and destination, for example, if the driver is heading to an airport y will be big, and if he's heading to a derelict suburb, y will be small, x will behave similarly.
That way, when a rider is heading to or located at a location that is popular among drivers, the value on his edges in the exchange network would be higher, that way, he'll be more likely to get a match and a higher price.