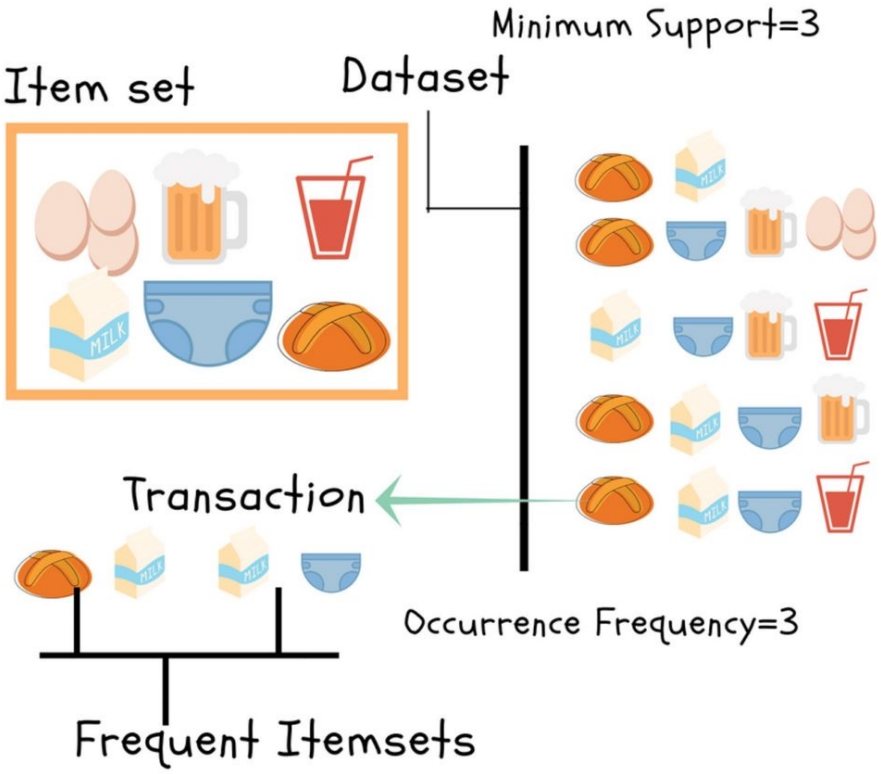
DATA SCIENCE MEI/1

University of Beira Interior
Department of Informatics

Hugo Pedro Proença,
hugomcp@di.ubi.pt, **2021/22**

Frequent Pattern Mining

- **Frequent patterns** are collections of items that appear in a dataset at a high frequency
- Such patterns reveal association rules and relations between variables.
- Frequent Pattern Mining is an important research area in data science applied to different domains:
  - Recommender systems;
  - Bioinformatics;
  - Decision making;



Source: http://www.dataception.in/blog/what-is-frequent-pattern-mining-association-and-how-does-it-support-business-analysis/

Frequent Pattern Mining

- Frequent patterns (a.k.a. Association Rule Mining) consists of first finding frequent item sets - satisfying a minimum support threshold, or percentage - from which strong association rules in the form A → B of are generated.
- These rules also satisfy a minimum confidence threshold (a prespecified probability of satisfying B under the condition that A is satisfied).
- Associations can be further analyzed to uncover correlation rules, which convey statistical correlations between item sets A and B.
- Various algorithms have been developed for frequent itemset mining, from which association and correlation rules are further derived. These algorithms can be classified into three categories:
  - Apriori-like algorithms;
  - Frequent pattern growth –based algorithms such as FP-growth;
  - Algorithms that use the vertical data format.

Apriori Algorithm

- The **Apriori** algorithm is the most classical algorithm for mining frequent item sets for Boolean (A -> B) association rules.
  - It was proposed by Agrawal and Srikant in 1994.
- It explores the level-wise mining property that all nonempty subsets of a frequent itemset must also be frequent.
- At the k^{th} iteration (k >= 2), it forms frequent candidate sets of k elements based on the frequent (k − 1)-item sets, and scans the database once to find the complete set of frequent k-item sets, L_k.
- Due to its popularity, several variations have been proposed, including hashing and transaction reduction, in order to make the procedure more efficient. Other variations include partitioning the data (mining on each partition and then combining the results) and sampling the data (mining on a data subset). These variations can reduce the number of data scans required to as little as two or even one.

Apriori Algorithm

Transaction	Items
A	1, 3, 4
B	2, 3, 5
C	1, 2, 3, 5
D	2, 5

Step 1: Count frequency of 1-item sets

Set	Frequency
1	2
2	3
3	3
4	1
5	3

Step 2: Remove the least frequent set

Set	Frequency
1	3
2	3
3	3
4	1
5	4

Apriori Algorithm

Transaction	Items
A	1, 3, 4
B	2, 3, 5
C	1, 2, 3, 5
D	2, 5

Step 3: Count frequency of 2-item sets (of remaining elements)

Set	Frequency
1, 2	1
1, 3	2
1, 5	1
2, 3	2
2, 5	3
3, 5	2

Step 4: Remove the least frequent set

Set	Frequency
1, 2	1
1, 3	2
1, 5	1
2, 3	2
2, 5	3
3, 5	2

Apriori Algorithm

Transaction	Items
A	1, 3, 4
B	2, 3, 5
C	1, 2, 3, 5
D	2, 5

Step 5: Count frequency of 3-item sets (of remaining elements)

Set	Frequency
2, 3, 5	2
1, 2, 3	0
1, 2, 5	0

Pseudocode:

```
Apriori(T, ε)
    L_1 ← {large 1 − itemsets}
    k ← 2
    while L_{k−1} is not empty
        C_k ← {c = Union(a, {b}) : a in L_{k−1} and b not in a, {s ⊆ c : |s| = k − 1} ⊆ L_{k−1}}
        for transactions t in T
            D_t ← {c in C_k : c ⊆ t}
            for candidates c in D_t
                count[c] ← count[c] + 1

        L_k ← {c in C_k : count[c] ≥ ε}
        k ← k + 1

    return Union(L_k)
```

Frequent Pattern Growth

- **Frequent Pattern Growth** is a method of mining frequent item-sets without candidate generation.
- This algorithm constructs a highly compact data structure (an FP-tree) to compress the original transaction database.
  - An FP-tree is a tree-like structure made of the initial item-sets of the database. The purpose of the FP tree is to mine the most frequent pattern. Each node of the FP tree represents an item of the itemset.
  - The root node represents *null* while the lower nodes represent the item-sets. The paths between the root and the leaves represent the item-sets
- Instead employing the generate-and-test strategy of Apriori-like methods, it focuses on frequent pattern (fragment) growth, which avoids costly candidate generation, and enables greater efficiency.

Frequent Pattern Growth

Transaction	Items
A	3, 4
B	2, 3, 5
C	1, 2, 3, 5
D	2, 5
E	5, 4
F	5, 3

Step 1: Count frequency of 1-item sets

Set	Frequency
1	1
2	3
3	4
4	2
5	5

Step 2: Sort 1-item sets in descending order

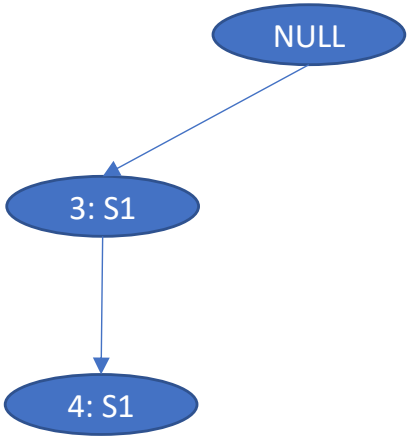
Set	Frequency
5	5
3	4
2	3
4	2
1	1

Frequent Pattern Growth

Transaction	Items
A	4, 3
B	2, 3, 5
C	1, 2, 3, 5
D	2, 5
E	5, 4
F	5, 3

Step 3: Insert the first ordered set {4, 3} → {3, 4}
Freq(3)=4, Freq(4)=2

As there are no nodes in the tree, new elements should be created and inserted

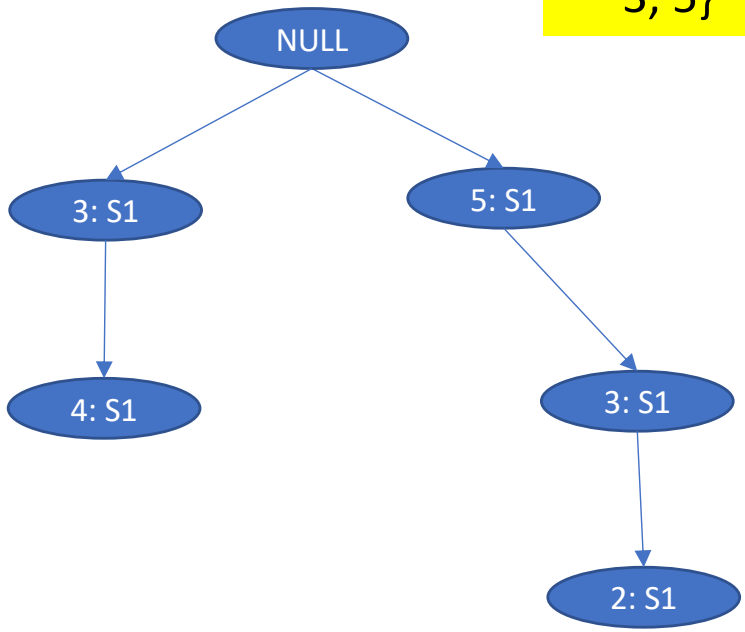


Step 4: Update Support

Frequent Pattern Growth

Transaction	Items
A	4, 3
B	2, 3, 5
C	1, 2, 3, 5
D	2, 5
E	5, 4
F	5, 3

Step 5: Insert the next ordered set {2, 3, 5} → {5, 3, 2}

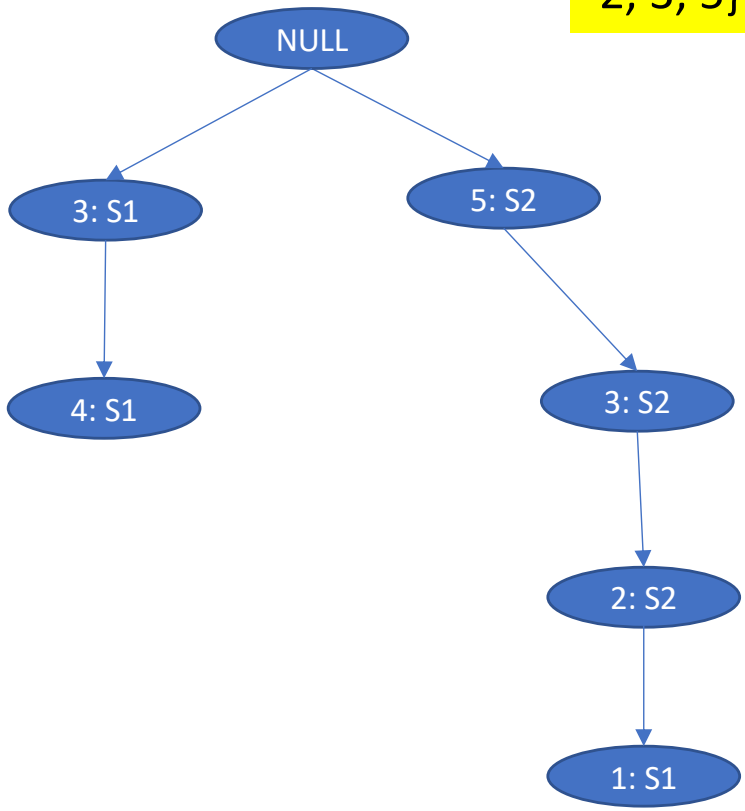


Step 6: Update Support

Frequent Pattern Growth

Transaction	Items
A	4, 3
B	2, 3, 5
C	1, 2, 3, 5
D	2, 5
E	5, 4
F	5, 3

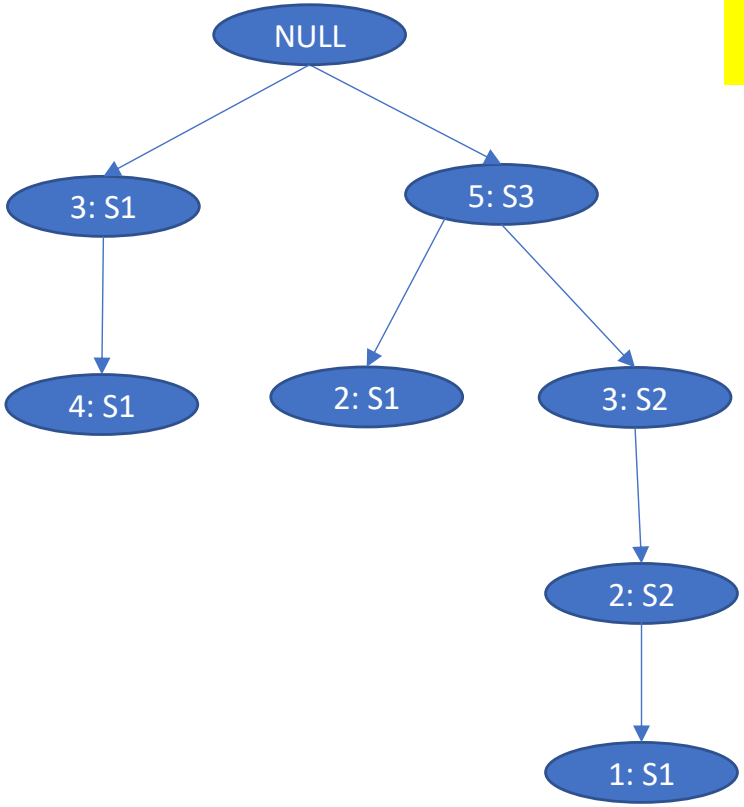
Step 7: Insert the next ordered set {1, 2, 3, 5} → {5, 3, 2, 1}



Step 8: Update Support

Frequent Pattern Growth

Transaction	Items
A	4, 3
B	2, 3, 5
C	1, 2, 3, 5
D	2, 5
E	5, 4
F	5, 3



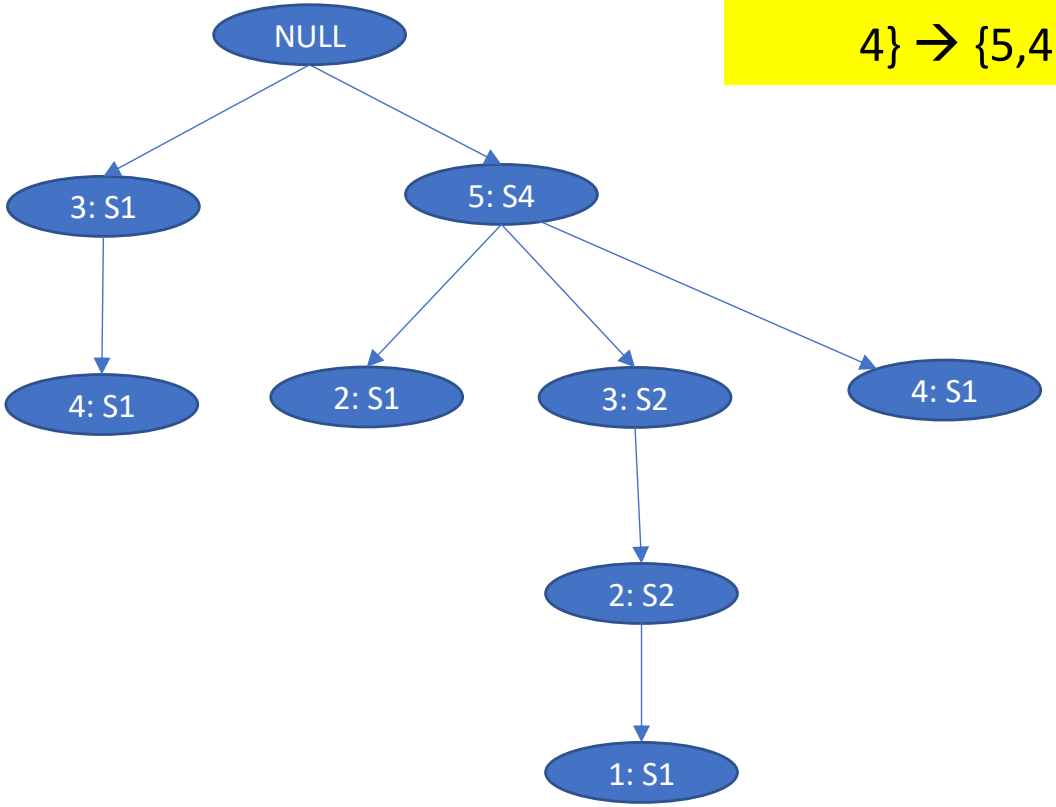
Step 9: Insert the next ordered set {2, 5} → {5,2} and {5}

Step 10: Update Support

Frequent Pattern Growth

Transaction	Items
A	4, 3
B	2, 3, 5
C	1, 2, 3, 5
D	2, 5
E	5, 4
F	5, 3

Step 11: Insert the next ordered set {5, 4} → {5,4}

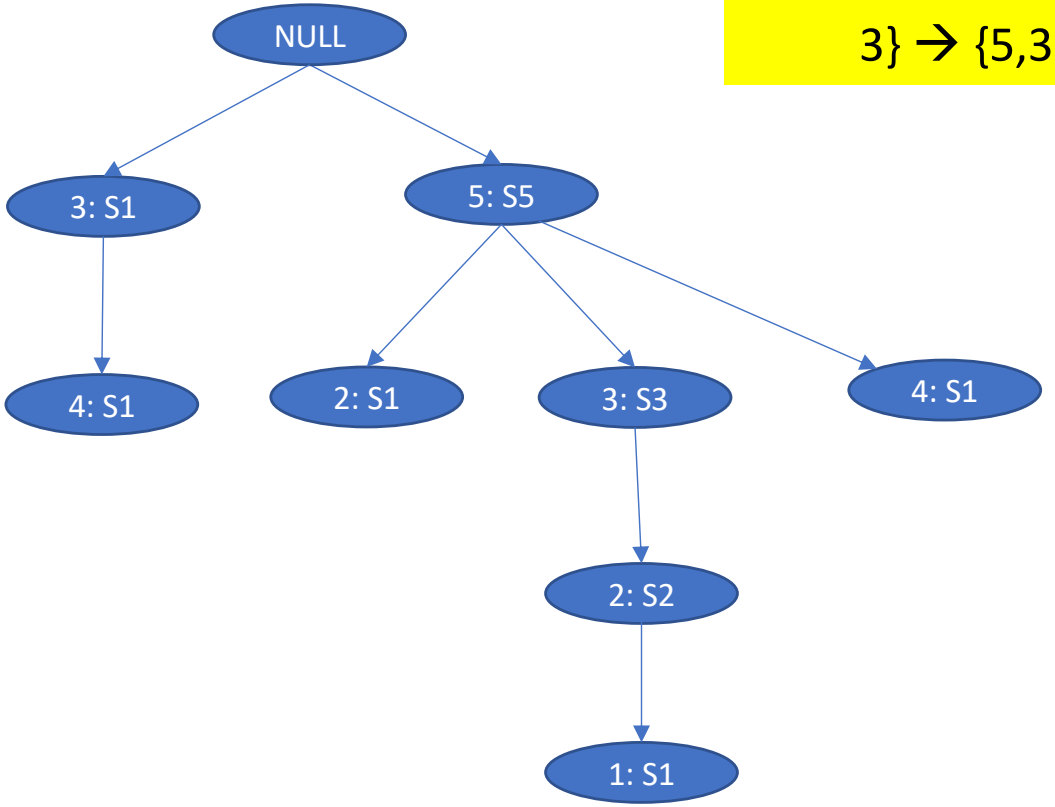


Step 12: Update Support

Frequent Pattern Growth

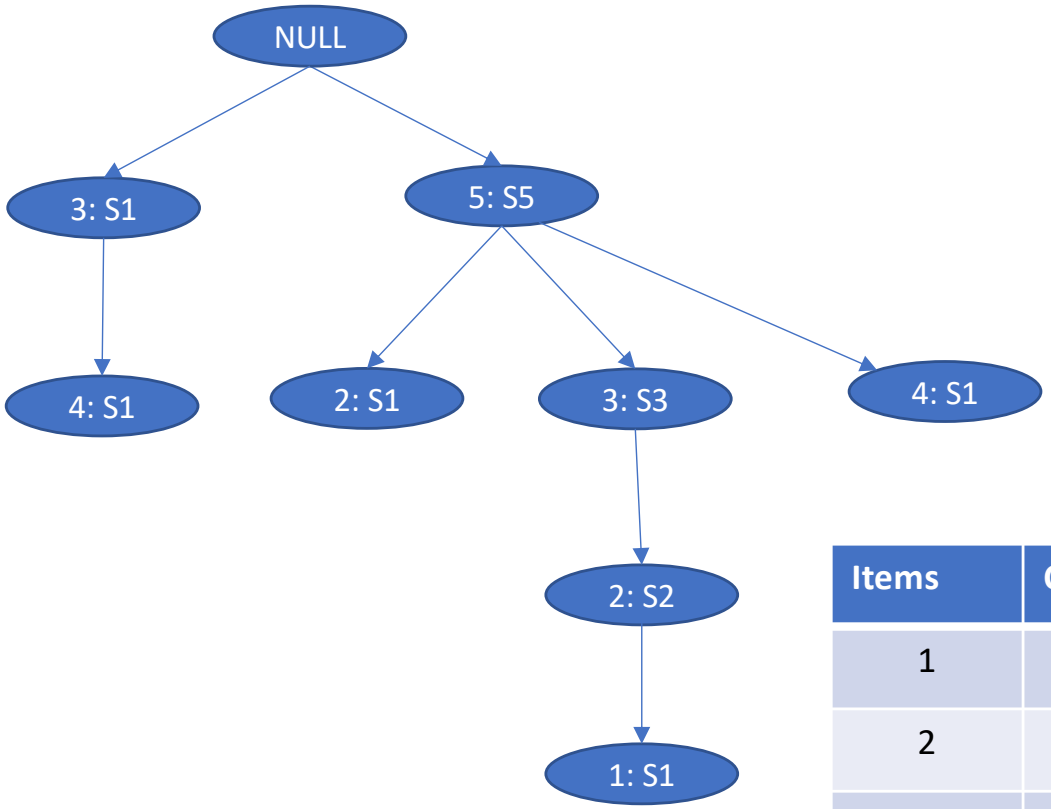
Transaction	Items
A	4, 3
B	2, 3, 5
C	1, 2, 3, 5
D	2, 5
E	5, 4
F	5, 3

Step 13: Insert the next ordered set {5, 3} → {5,3}



Step 14: Update Support

Frequent Pattern Growth



Step 8: Build the Conditional Pattern Base, which are the paths that lead to each item

Items	Conditional Pattern Base
1	{5, 3, 2: S1}
2	{5: S1}, {5, 3: S2}
3	{5: S3}
4	{3: S1}, {5: S1}
5	

Frequent Pattern Growth

Items	Conditional Pattern Base
1	{5, 3, 2: S1}
2	{5: S5}, {5, 3: S2}
3	{5: S3}
4	{3: S1}, {5: S1}
5	

Step 11: Build the **Conditional Frequent Pattern Tree**, that identifies the common elements in each row of the Conditional Pattern Base and sums their supports

Items	Conditional Pattern Base	Conditional Frequent Pattern Tree
1	{5, 3, 2: S1}	{5, 2, 3: S1}
2	{5: S1}, {5, 3: S2}	{5: S3}, {3: S2}
3	{5: S3}	{5: S3}
4	{3: S1}, {5: S1}	{3: S1}, {5: S1}
5		

Frequent Pattern Growth

Items	Conditional Pattern Base	Conditional Frequent Pattern Tree
1	{5, 3, 2: S1}	{5, 2, 3: S1}
2	{5: S1}, {5, 3: S2}	{5: S3}, {3: S2}
3	{5: S3}	{5: S3}
4	{3: S1}, {5: S1}	{3: S1}, {5: S1}
5		

Items	Frequent Patterns
1	{5, 2, 3, **1**: S1}
2	{5, **2**: S3}, {3, **2**: S2}
3	{5, **3**: S3}
4	{3, **4**: S1}, {5, **4**: S1}
5	

Step 12: From the Conditional Frequent Pattern Trees (CFPT), the Frequent Patterns are detected by pairing each CFPT element, to the item

Frequent Pattern Mining Exercise

- Consider the following set of transactions of a supermarket.

- Find, according to the **APRIORI** and the **FP-GROWTH** algorithms, the Frequent Patterns in such dataset, considering:
  - Support = 20%
  - Support = 50%

Transaction	Items
A	1, 7, 4, 6,
B	2, 3, 5
C	1, 2, 3, 5
D	4, 6, 7
E	8, 7, 2, 3
F	2, 3, 5, 2, 1
G	4, 5, 6, 7, 9
H	9, 1, 3, 2, 4
I	4, 6, 7, 5
J	7, 9, 5, 6, 1