1 So far

2 The Cowherd of Gokul
The story so far ...

- We have seen various control flows.
- We have seen multi-dimensional arrays and the `char` data type.
- We saw the use of functions and calling methods.
- We have seen structs, sorting, searching.

This week...

A real life problem..
Srirang is a cowherd from Gokul. He has a single cow. By god’s grace:

- The cow gives 50 litres of milk everyday.
- The expense of maintaining this cow is Rs. 250 per day.

Srirang wishes to sell this milk. Every evening, Srirang gets bids from various parties. Each bid is of the form:

- Name of the bidder.
- The price at which he/she will purchase milk.
- The volume that he/she requires.
Srirang is a cowherd from Gokul. He has a single cow. By god’s grace:
- The cow gives **50 litres** of milk everyday.
- The expense of maintaining this cow is **Rs. 250** per day.

Srirang wishes to sell this milk. Every evening, Srirang gets bids from various parties. Each bid is of the form:
- Name of the bidder.
- The **price** at which he/she will purchase milk.
- The **volume** that he/she requires.

Looking at the bids, Srirang decides on a price for the next day, say $X$. This price is offered to all customers. The customers who can afford the price collect the milk and pay **Rs. $X$/litre**.

Here is an example:

<table>
<thead>
<tr>
<th>name</th>
<th>volume</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>roshni</td>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>prema</td>
<td>15</td>
<td>8</td>
</tr>
<tr>
<td>radha</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>rukmi</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>gauri</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>neha</td>
<td>10</td>
<td>6</td>
</tr>
</tbody>
</table>
He fixes a price of Rs.5. Gauri goes away. There is an overall demand of 60. The others distribute the supply of 50 liters somehow. Sriang earns Rs. 250.
He fixes a price of **Rs.5. Gauri** goes away. There is an overall demand of **60.** The others distribute the supply of **50 liters** somehow. Sriang earns **Rs. 250.**

He gets a bit greedy and fixes the price to **Rs. 7** and makes the following table:

<table>
<thead>
<tr>
<th>Declared Price</th>
<th>Demand</th>
<th>Supply</th>
<th>Earnings</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>40</td>
<td>40</td>
<td>280</td>
</tr>
</tbody>
</table>
Question: What price Rs. X/liter should Srirang set to maximize his profits?
Question: What price Rs. $X$/liter should Srirang set to maximize his profits?

Some Observations:

- Clearly as $X$ increases, the demand decreases.
- For the price $X$ if the demand is greater than 50 then the supply can only be 50.
- For the price $X$ if the demand is less than 50 then it can be met.
- We need to maximize $X \times \text{Supply}$. 
The Poser

**Question:** What price Rs. X/liter should Srirang set to maximize his profits?

**Some Observations:**
- Clearly as $X$ increases, the demand decreases.
- For the price $X$ if the demand is greater than 50 then the supply can only be 50.
- For the price $X$ if the demand is less than 50 then it can be met.
- We need to maximize $X \times \text{Supply}$.

**Key Observation:**
- Clearly the optimum $X$ is a price offered by some customer.
Question: What price Rs. $X$/liter should Srirang set to maximize his profits?

Some Observations:
- Clearly as $X$ increases, the demand decreases.
- For the price $X$ if the demand is greater than 50 then the supply can only be 50.
- For the price $X$ if the demand is less than 50 then it can be met.
- We need to maximize $X \times \text{Supply}$.

Key Observation:
- Clearly the optimum $X$ is a price offered by some customer.

Why is this?
- The net earning depends on the demand.
- If, for prices $X_1 < X_2$, the demand is unchanged then clearly $X_2$ is preferred.
- The demand can only change when we hit a customer price.
Solution

A computational solution is now easy:

- **Try every customer price.**
- Compute Demand at that price.
- Compute Supply and Earnings.
- **Select the best!**
Solution

A computational solution is now easy:

- **Try every customer price.**
- Compute Demand at that price.
- Compute Supply and Earnings.
- Select the best!

Data required:

<table>
<thead>
<tr>
<th>The bids</th>
<th>The basic data structures are:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Maximum Supply (50L)</td>
<td>struct bid</td>
</tr>
<tr>
<td>My Costs (Rs. 250)</td>
<td>`{</td>
</tr>
<tr>
<td></td>
<td>char name[6];</td>
</tr>
<tr>
<td></td>
<td>int price, vol;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>bid bidlist[10]</td>
</tr>
<tr>
<td></td>
<td>int MaxSupply;</td>
</tr>
</tbody>
</table>
Solution

A computational solution is now easy:

- Try every customer price.
- Compute Demand at that price.
- Compute Supply and Earnings.
- Select the best!

Data required:

<table>
<thead>
<tr>
<th>The bids</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Maximum Supply (50L)</td>
</tr>
<tr>
<td>My Costs (Rs. 250)</td>
</tr>
</tbody>
</table>

The basic data structures are:

```c
struct bid
{
    char name[6];
    int price, vol;
}
```

```c
bid bidlist[10]
int MaxSupply;
```

The basic functions are:

```c
int ComputeDemand (bid bidlist[], int price);
int Supply;
Supply = Min(MaxSupply, Demand);
```
int ComputeDemand(bid bidlist[], int X, int N)
{
    int i, d = 0;
    for (i = 0; i < N; i = i + 1)
        if (bidlist[i].price >= X)
            d = d + bidlist[i].volume;
    return (d);
}
Compute Demand

```c
int ComputeDemand(bid bidlist[], int X, int N)
{
    int i, d=0;
    for (i=0; i<N; i++)
        if (bidlist[i].price >= X)
            d = d + bidlist[i].volume;
    return (d);
}
```

What's happening?

- \(X\) is the price.
- \(N\) is the number of bids.
- \(d\) is the total.

We rush through all the bids, and total up all demands greater than or equal to \(X\).
int main()
{
    int i,N,MaxSupply,E,Earnings,Xbest;
    int X,demand, supply, Sup; bid bids[20];
    cout << " N and MaxSupply? \n";
    cin >> N >> MaxSupply;
    for (i=0;i<N;i=i+1)
    {
        cin >> bids[i].name >> bids[i].volume >> bids[i].price;
    }
    Xbest=0;
    Earnings=0;
    Sup=0;
    IMPORTANT CODE HERE
    cout << "best price " << Xbest << "\n";
    cout << "Earnings " << Earnings << "\n";
    cout << "Supply " << Sup << "\n";
};
The important part

Xbest=0;
Earnings=0;
Sup=0;
for (i=0;i<N;i=i+1)
{
    X=bids[i].price;
    demand=ComputeDemand(bids, X,N);
    supply=min(demand,MaxSupply);
    E=supply*X;
    if (E>Earnings)
    {
        Earnings=E;
        Xbest=X;
        Sup=supply;
    }
}; // of for

Whats happening:

- Keep
  | Xbest | the best price so far |
  | E     | earnings at Xbest    |
  | Sup   | supply at that price |

- Initialize this data, and run across each price. This is because we know that the optimum occurs at some offered price.

- Update the variables above for each price. Call ComputeDemand to do this.
Input and Output

6 50
roshni 5 20
prema 15 8
radha 20 10
rukmi 10 5
gauri 10 3
neha 10 6

Thus maximum supply is 50 and there are 6 bids.
Input and Output

Thus maximum supply is 50 and there are 6 bids.

[sohoni@nsl-13 lectures]$ ./a.out
N and MaxSupply?
best price 8
Earnings 320
Supply 40

Thus we see that the best price is 8 and that the supply at this price is 40 litres. Earnings are Rs. 320.
Input and Output

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>50</td>
</tr>
<tr>
<td>roshni</td>
<td>5 20</td>
</tr>
<tr>
<td>prema</td>
<td>15 8</td>
</tr>
<tr>
<td>radha</td>
<td>20 10</td>
</tr>
<tr>
<td>rukmi</td>
<td>10 5</td>
</tr>
<tr>
<td>gauri</td>
<td>10 3</td>
</tr>
<tr>
<td>neha</td>
<td>10 6</td>
</tr>
</tbody>
</table>

Thus maximum supply is 50 and there are 6 bids.

[sohoni@ns1-13 lectures]$ ./a.out N and MaxSupply?

best price 8
Earnings 320
Supply 40

Thus we see that the best price is 8 and that the supply at this price is 40 litres. Earnings are Rs. 320. It is curious that:

- Gauri is refused, and yet..
Input and Output

6  50
roshni 5  20
prema 15 8
radha 20 10
rukmi 10 5
gauri 10 3
neha 10 6

Thus maximum supply is 50 and there are 6 bids.

[sohoni@nsl-13 lectures]$ ./a.out
N and MaxSupply?
best price 8
Earnings  320
Supply  40

Thus we see that the best price is 8 and that the supply at this price is 40 litres. Earnings are Rs. 320. It is curious that:

- Gauri is refused, and yet..
- 10 litres of milk is left behind!
### Input and Output

Thus maximum supply is 50 and there are 6 bids.

6 50
roshni 5 20
prema 15 8
radha 20 10
rukmi 10 5
gauri 10 3
neha 10 6

Thus we see that the **best price** is 8 and that the supply at this price is 40 litres. Earnings are Rs. 320. It is curious that:

- **Gauri** is refused, and yet..
- 10 litres of milk is left behind!
- So much for MARKET ECONOMY!
Two questions

What if there were 1000 bids?

- There are 1000 possible prices $X$. Thus the outer loop will run 1000 times. In other words, $\text{ComputeDemand}$ is called 1000 times.
- Each call of $\text{ComputeDemand}$ will take 1000 steps!
- Thus the time taken is $1000^2$. In other words, this is an $O(N^2)$ algorithm.

Can anything be done?
Two questions

What if there were 1000 bids?

- There are 1000 possible prices $X_i$. Thus the outer loop will run 1000 times. In other words, `ComputeDemand` is called 1000 times.
- Each call of `ComputeDemand` will take 1000 steps!
- Thus the time taken is $1000^2$. In other words, this is an $O(N^2)$ algorithm.

Can anything be done?

- Sort the bids in decreasing order. This takes $O(N \log N)$ time.
- Certainly
- Eliminate `ComputeDemand`.
- Demand $D_i$ at price $X_i$ is the demand at $X_{i-1}$ plus the volume $V_i$.

$$D_i = D_{i-1} + V_i$$
Two questions

What if there were 1000 bids?

- There are 1000 possible prices \( X \). Thus the outer loop will run 1000 times. In other words, \texttt{ComputeDemand} is called 1000 times.
- Each call of \texttt{ComputeDemand} will take 1000 steps!
- Thus the time taken is \( 1000^2 \). In other words, this is an \( O(N^2) \) algorithm.

Can anything be done?

- Sort the bids in decreasing order. This takes \( O(N \log N) \) time.
- Eliminate \texttt{ComputeDemand}.
- Demand \( D_i \) at price \( X_i \) is the demand at \( X_{i-1} \) plus the volume \( V_i \).

\[
D_i = D_{i-1} + V_i
\]

**Assignment**

Implement \texttt{sortedsrirang.cpp}
The second question

Siddhartha is Srirang’s older brother. He gets

- **buy bids** just as Srirang, but also

- **sell bids**.

<table>
<thead>
<tr>
<th>name</th>
<th>volume</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>roshni</td>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>prema</td>
<td>15</td>
<td>8</td>
</tr>
<tr>
<td>radha</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>rukmi</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>gauri</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>neha</td>
<td>10</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>volume</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>srirang</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>gopal</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>vithal</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>narayan</td>
<td>15</td>
<td>6</td>
</tr>
</tbody>
</table>
Siddhartha is Srirang’s older brother. He gets
• **buy bids** just as Srirang, but also
• **sell bids**.

<table>
<thead>
<tr>
<th>name</th>
<th>volume</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>roshni</td>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>prema</td>
<td>15</td>
<td>8</td>
</tr>
<tr>
<td>radha</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>rukmi</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>gauri</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>neha</td>
<td>10</td>
<td>6</td>
</tr>
</tbody>
</table>

Siddhartha must announce
• a buying price $Y$ at which he will buy milk.
• a selling price $X$ at which he will sell milk.

Write a program to compute the best pair $(Y,X)$ which maximizes his earnings.
The second question

Siddhartha is Srirang’s older brother. He gets
- **buy bids** just as Srirang, but also
- **sell bids**.

<table>
<thead>
<tr>
<th>name</th>
<th>volume</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>roshni</td>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>prema</td>
<td>15</td>
<td>8</td>
</tr>
<tr>
<td>radha</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>rukmi</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>gauri</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>neha</td>
<td>10</td>
<td>6</td>
</tr>
</tbody>
</table>

Siddhartha must announce
- a buying price $Y$ at which he will buy milk.
- a selling price $X$ at which he will sell milk.

Write a program to compute the best pair $(Y, X)$ which maximizes his earnings.

Does this MARKET function any better?
The second question

Siddhartha is Srirang’s older brother. He gets

- **buy bids** just as Srirang, but also
- **sell bids**.

<table>
<thead>
<tr>
<th>name</th>
<th>volume</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>roshni</td>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>prema</td>
<td>15</td>
<td>8</td>
</tr>
<tr>
<td>radha</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>rukmi</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>gauri</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>neha</td>
<td>10</td>
<td>6</td>
</tr>
</tbody>
</table>

Siddhartha must announce

- a **buying price** $Y$ at which he will buy milk.
- a **selling price** $X$ at which he will sell milk.

Write a program to compute the best pair $(Y,X)$ which maximizes his earnings.

Does this MARKET function any better?

**NO**