At time, a better **upper bound** instead of use n*cost(wc) is used **Amortized Complexity**

cost(wc) : worst time for each single task

n*cost(wc) Traditional upper bound

Amortized Complexity is the amount you want to charge the task

n*(amortized cost of task)

In tradition upper bound, you want to charge each task >= cost more than the actual one. In amortized , you can charge <cost than the actual cost

**Amortized Rule (**Rule that must be hold after you assign amortized cost**):**

1. sum(actual cost)<=sum(Amortized Cost)

**Potential Function**

- P(i)=Amortized cost(i)-Actual cost(i)+P(i-1) =>P defination
- sum(p(i)-p(i-1))=sum(Amortized cost(i)-Actual cost(i))
- p(n)-p(0)=sum(Amortized cost(i)-Actual cost(i))>=0