CS4311
Design and Analysis of Algorithms

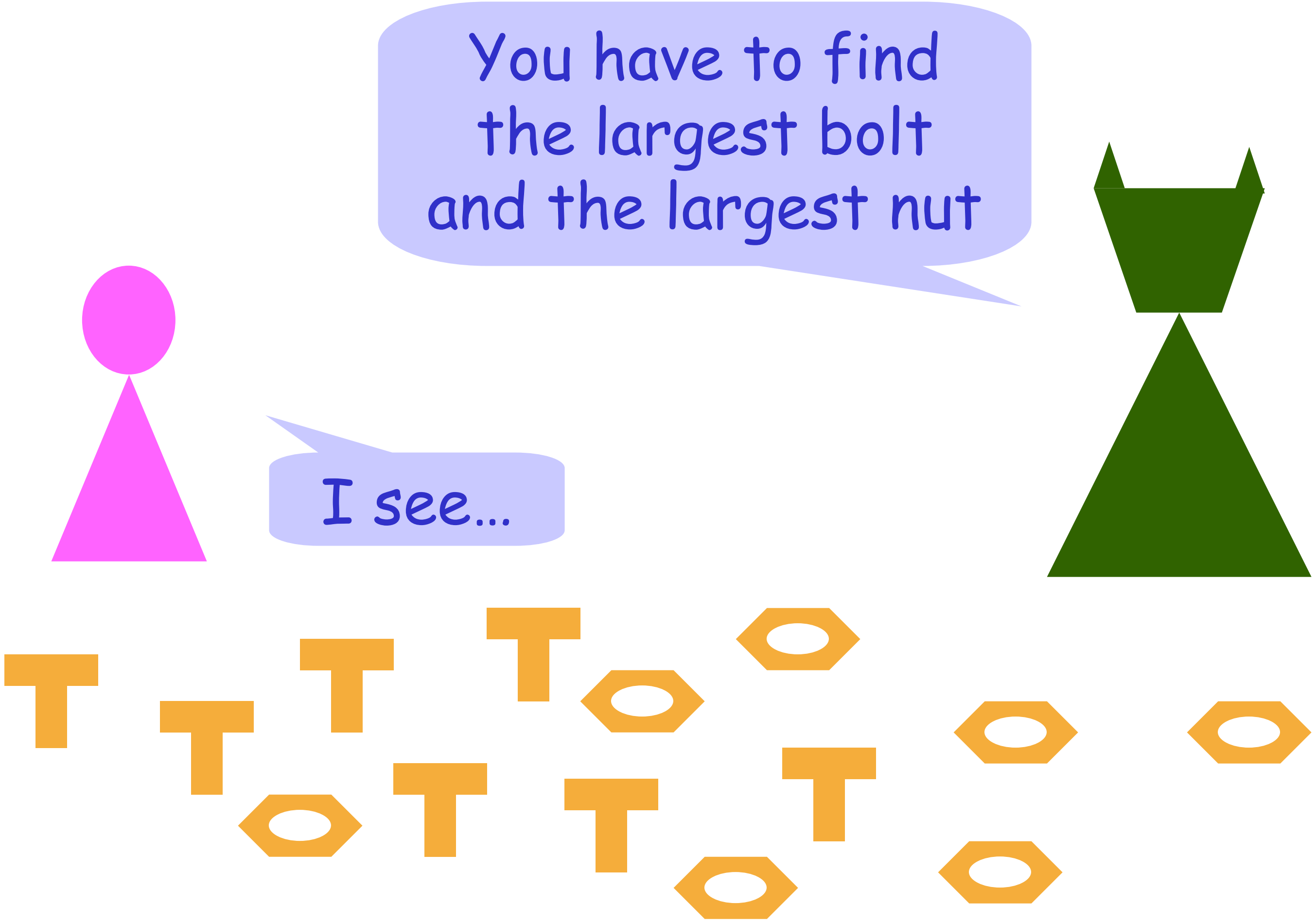
Tutorial: Randomized Selection

About this tutorial

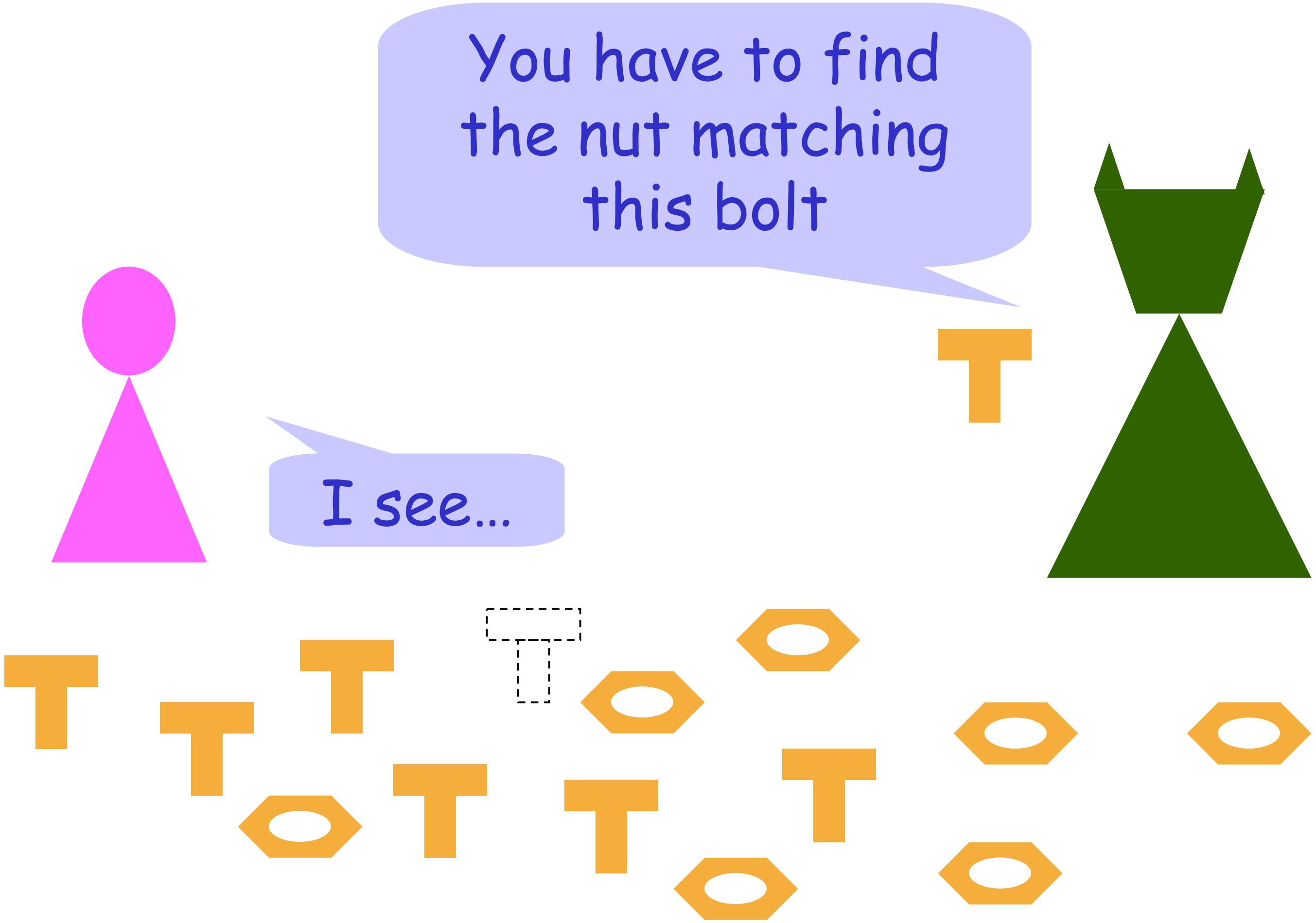
- Cinderella's New Problem

- Randomized Selection
  - Modification of Quicksort
  - Average-Case

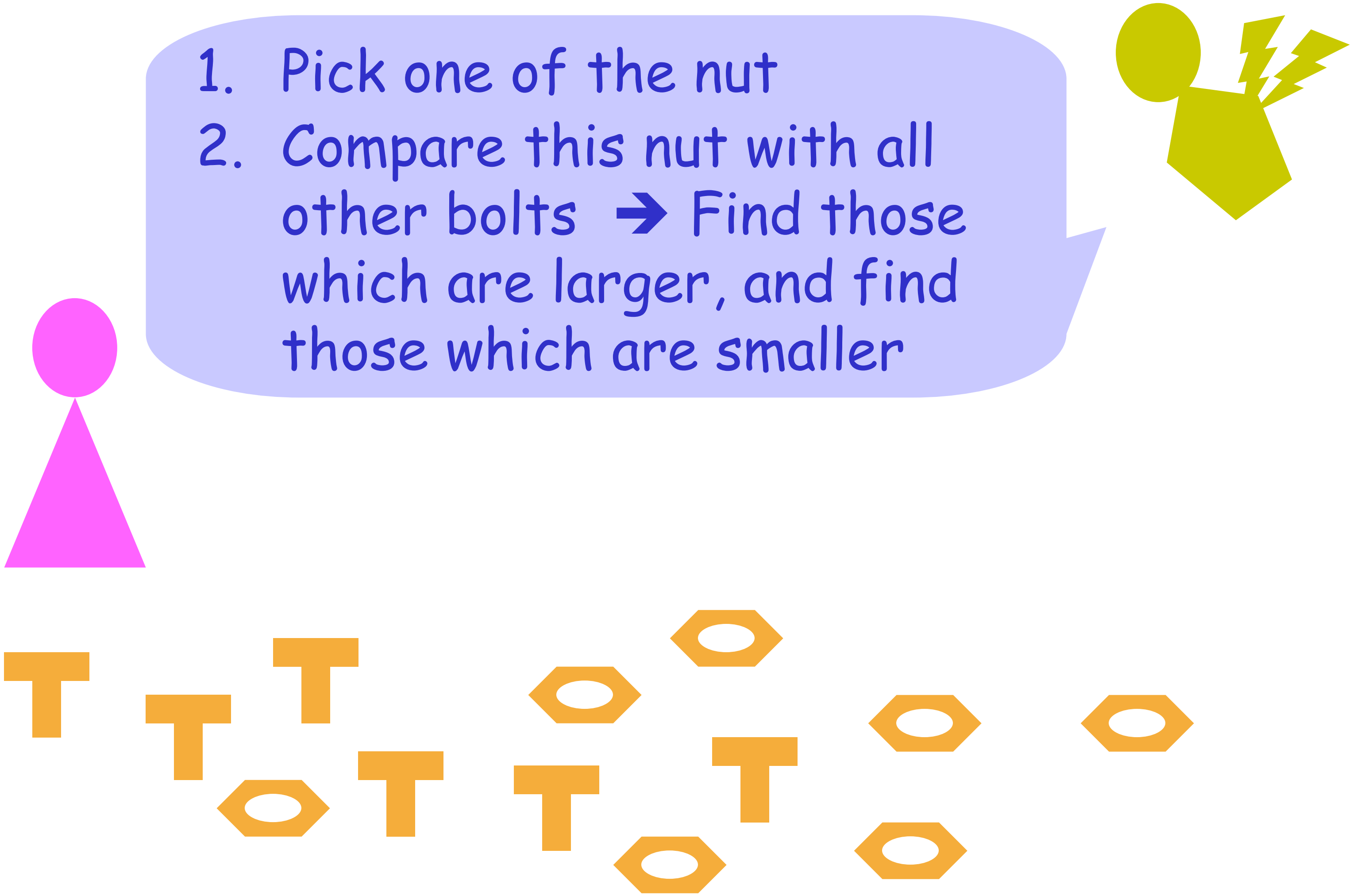
Cinderella's Problem



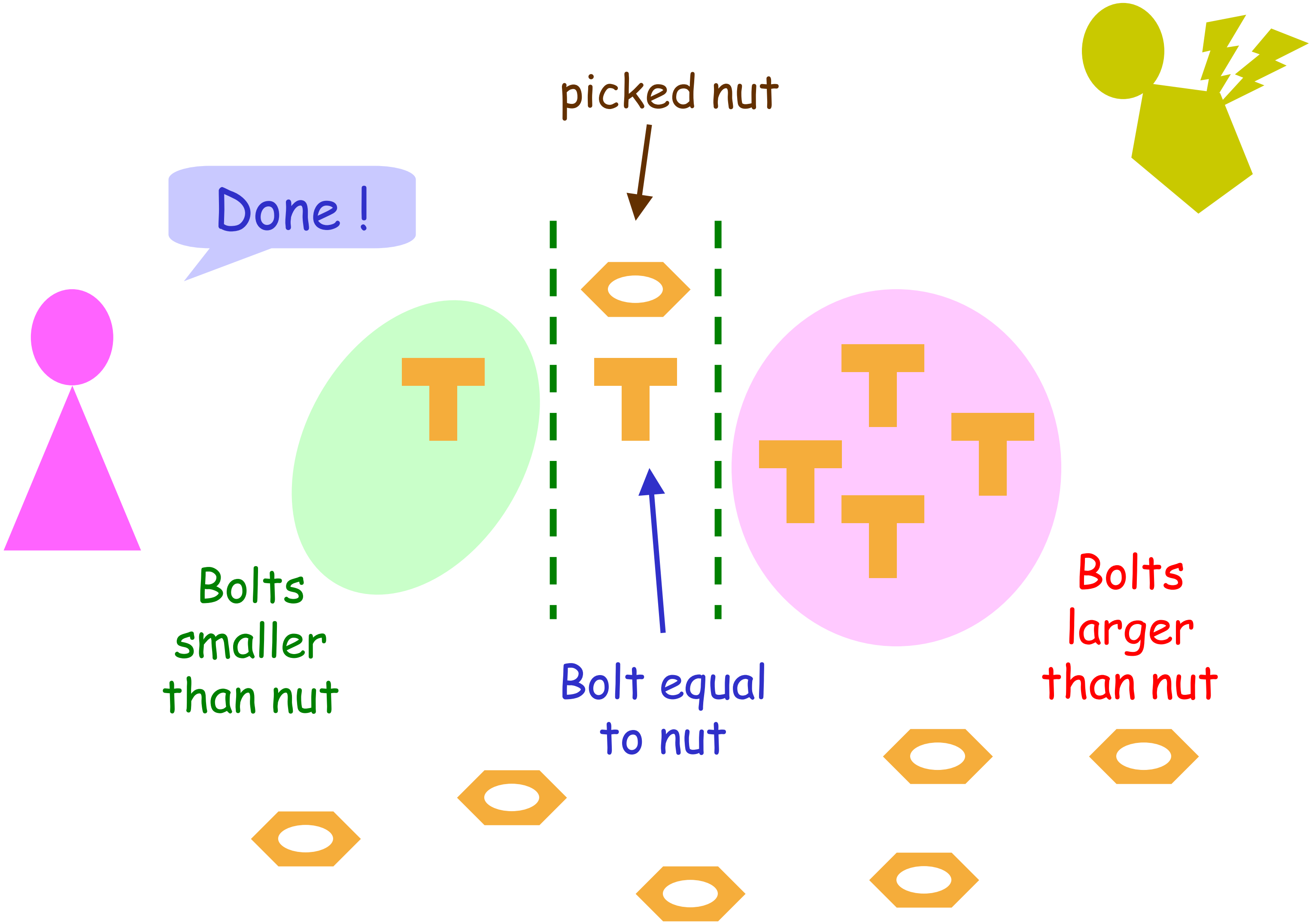
Cinderella's New Problem



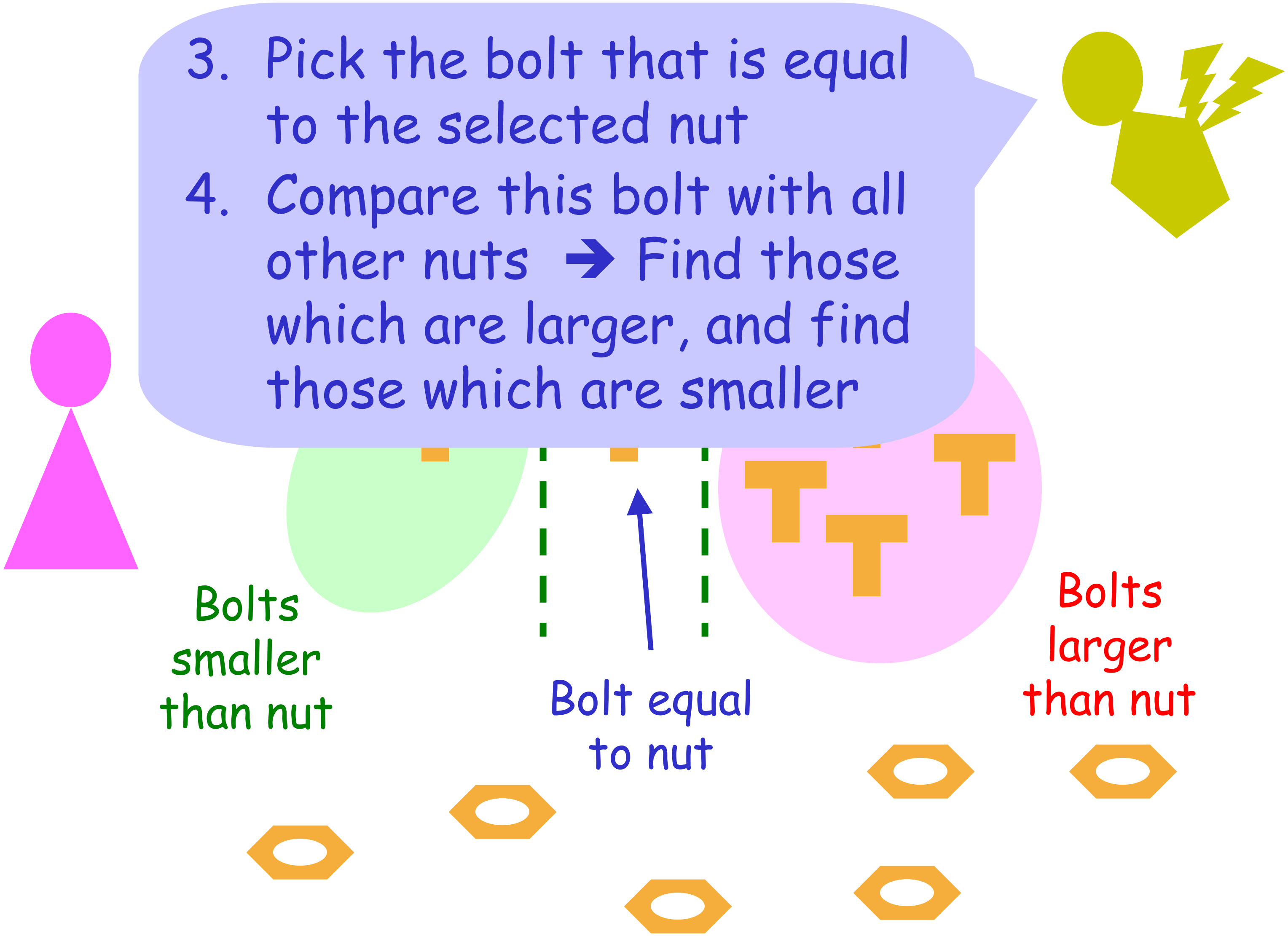
Fairy Godmother's Proposal



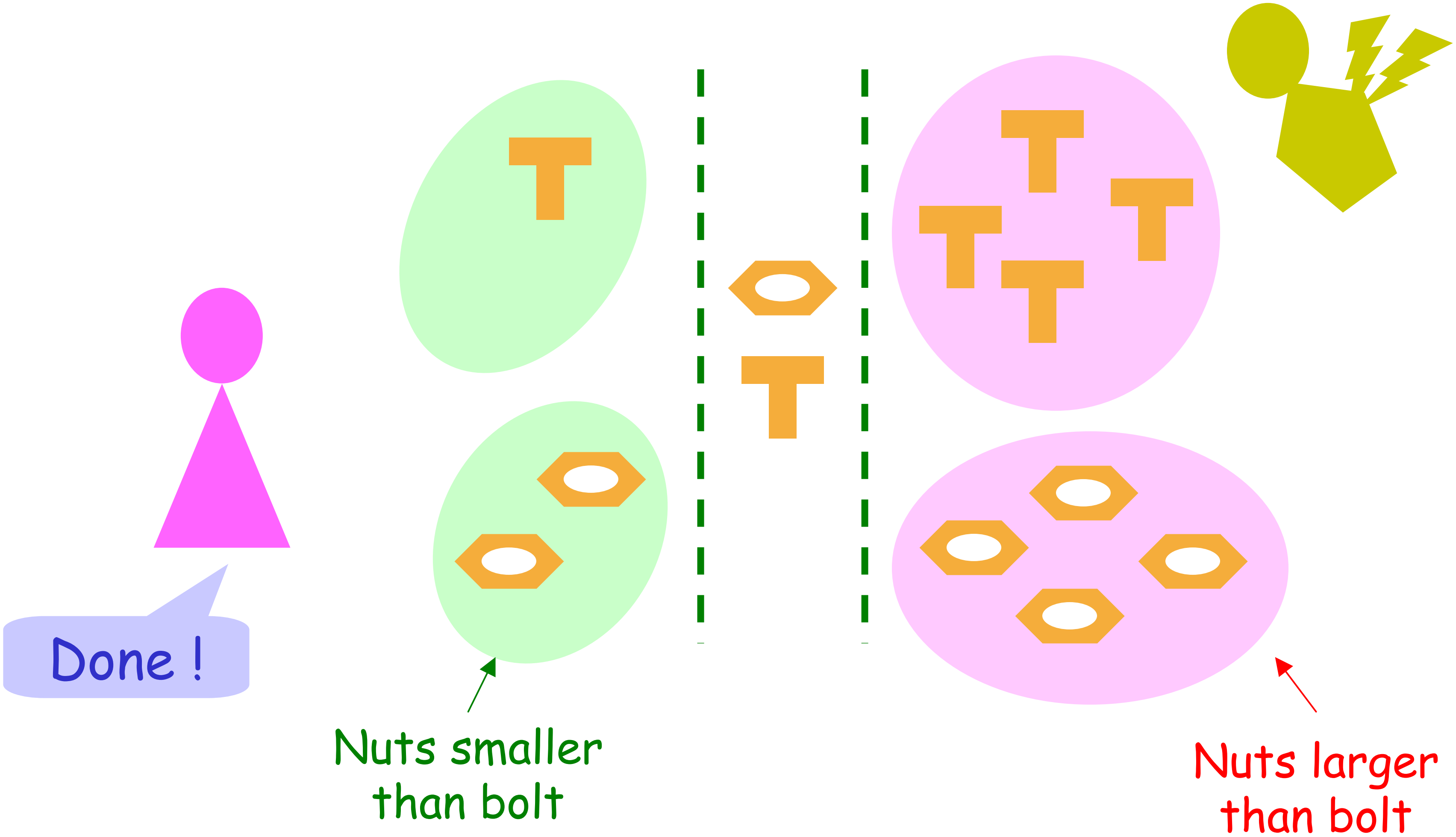
Fairy Godmother's Proposal



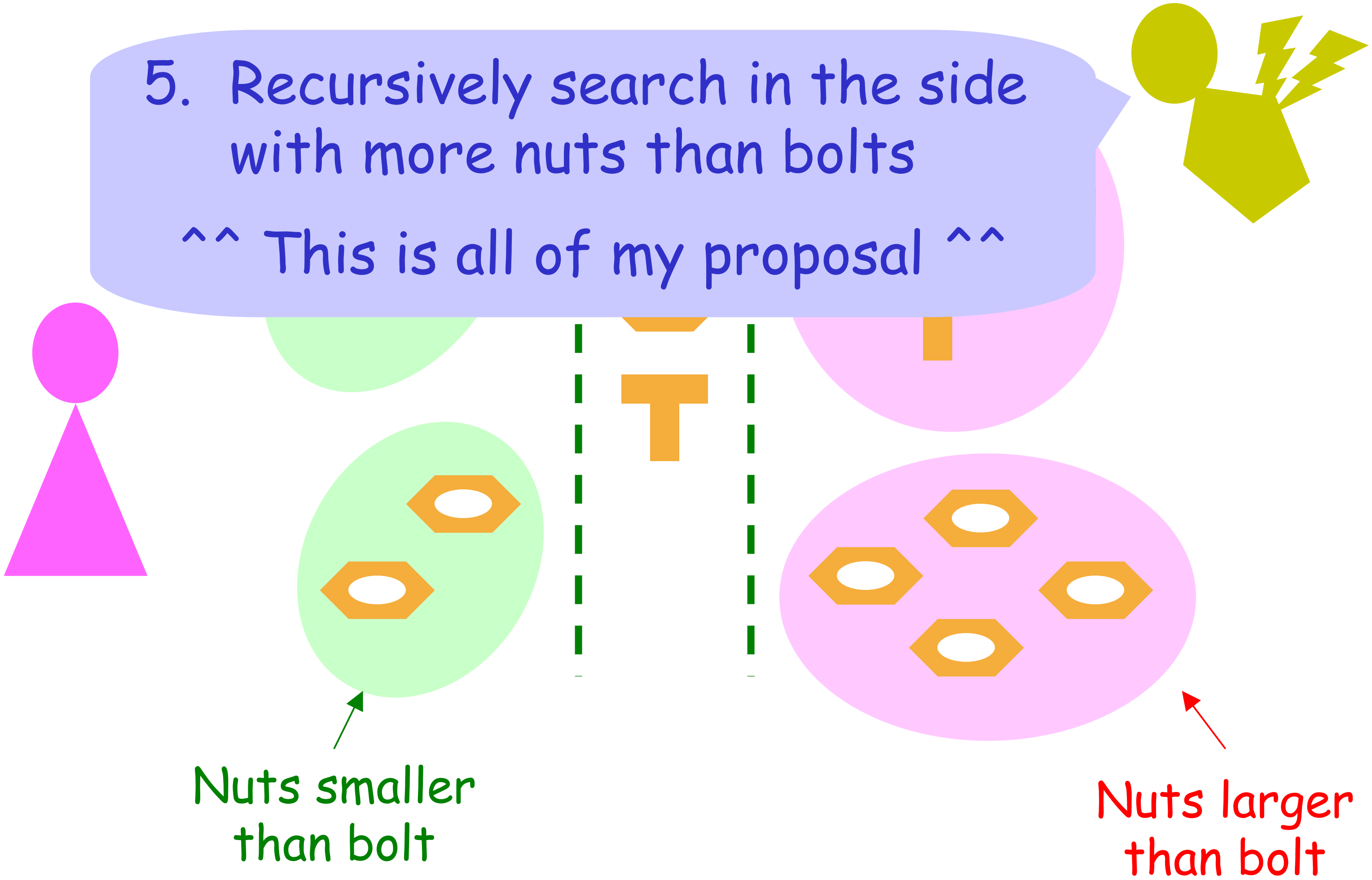
Fairy Godmother's Proposal



Fairy Godmother's Proposal



Fairy Godmother's Proposal



Fairy Godmother's Proposal

- Can you see why Fairy Godmother's proposal is a correct algorithm?
- What is the running time ?
  - Worst-case: $\Theta(n^2)$ comparisons
  - No better than the brute force approach !!
- Though worst-case runs badly, the average case is good: $\Theta(n)$ comparisons

Review: Quicksort

The Quicksort algorithm works as follows:

```
Quicksort(A,p,r)      /* to sort array A[p..r] */
   1. if ( p ≥ r )  return;
   2. q = Partition(A,p,r);
   3. Quicksort(A, p, p+q-1);
   4. Quicksort(A, p+q+1, r);
```

To sort A[1..n], we just call Quicksort(A,1,n)

Our Algorithm

The previous algorithm works as follows:

```
/* Search in array A[p..r], knowing that the
   desired value is inside A[p..r]          */
Search(A,p,r)
   1. if ( p ≥ r )   return;
   2. q = Partition(A,p,r);
   3. If target is in A[p..p+q-1]
              Search(A, p, p+q-1) ;
   4. Else Search(A, p+q+1, r);
```

Average Running Time

- The previous algorithm is called a selection algorithm, which allows us to find out the kth smallest item, for any k

- What is the average running time ?

  Let $T(n)$ denote average running time on input of size n

  We shall show that $T(n) = O(n)$

Average Running Time

Inductive Case (assume n is even):

$$T(n) \leq \Sigma_q\ (1/n) \max\big(T(q), T(n-q-1)\big) + \Theta(n)$$

$$\leq (2/n)\ \Sigma_{q=1 \text{ to } n/2}\ T(n-q-1) + \Theta(n)$$

$$\leq (2/n)\ c(3n/2-1)(n/2)/2 + \Theta(n)$$

$$= (3/4)cn + \Theta(n)$$

$$\leq cn$$ when c is large enough

For odd n, we get $T(n) \leq (3/4)cn + (1/2)c + \Theta(n)$

Average Running Time

Conclusion: $T(n) = O(n)$

- In fact, there is another proof which uses a similar technique as we use in Quicksort

Average Running Time

Let X = # comparisons in all Partition

Then, we have:

Running time = O(n + X)

varies on input

Finding average of X (i.e. #comparisons)
gives average running time

Average # of Comparisons

Recall the notation:

- Let $a_1, a_2, \ldots, a_n$ denote the set of n numbers initially placed in the array
- Further, assume $a_1 < a_2 < \ldots < a_n$
- Let X_{ij} = # comparisons between a_i and a_j in all Partition calls

Average # of Comparisons

Then, X = # comparisons in all Partition calls

$$= X_{12} + X_{13} + \ldots + X_{n-1,n}$$

➔ Average # comparisons :

$$E[X] = E[X_{12} + X_{13} + \ldots + X_{n-1,n}]$$

$$= E[X_{12}] + E[X_{13}] + \ldots + E[X_{n-1,n}]$$

Later, we shall group $E[X_{ij}]$ terms properly, so that we can easily show $E[X] = O(n)$

Comparison between a_i and a_j

Question: # times a_i be compared with a_j ?

Answer: At most once, which happens only if a_i or a_j are chosen as pivot

1	3	2	4	5	6	8	7

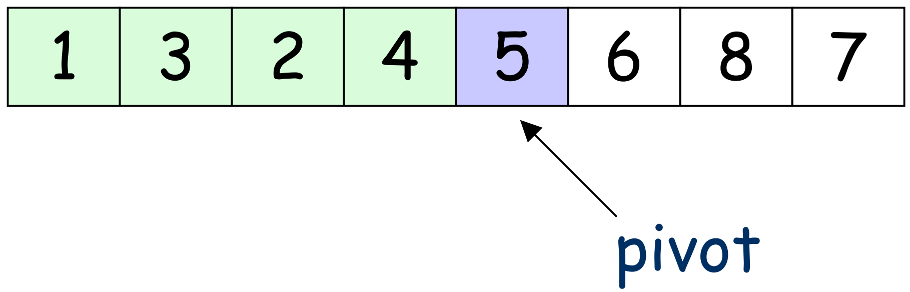
pivot

After that, the pivot is fixed and is never compared with the others

Comparison between a_i and a_j

Question: Will a_i always be compared with a_j ?

Answer: No. E.g., 4 and 6 are not compared



- In addition, if target is the 6^{th} smallest item, then 2 and 4 are also not compared
- When will a comparison occur ?