WORKED EXAMPLE 2.1

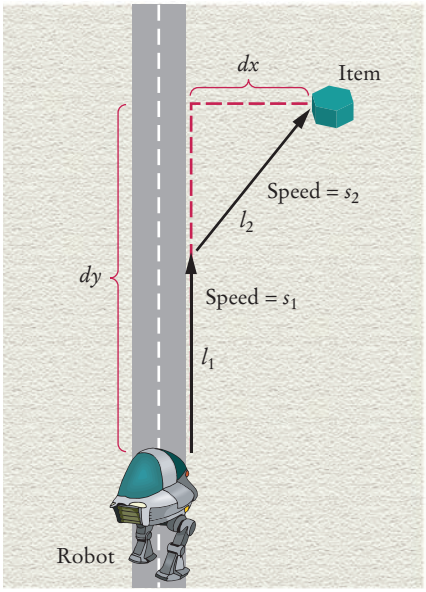
Computing Travel Time

In this example, we develop a hand calculation to compute the time that a robot requires to retrieve an item from rocky terrain.

A robot needs to retrieve an item that is located in rocky terrain adjacent to a road. The robot can travel at a faster speed on the road than on the rocky terrain, so it will want to do so for a certain distance before moving on a straight line to the item.

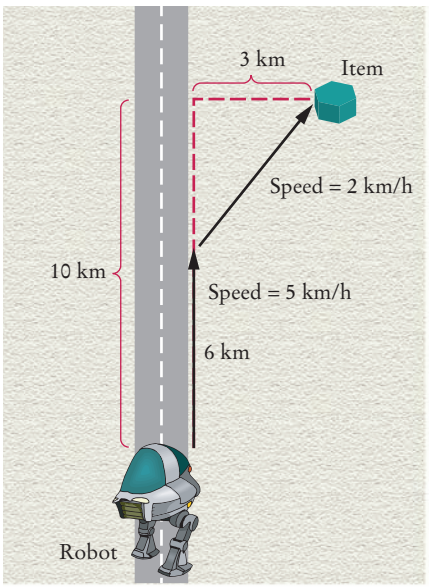
Your task is to compute the total time taken by the robot to reach its goal, given the following inputs:

- The distance between the robot and the item in the x- and y-direction (dx and dy)
- The speed of the robot on the road and the rocky terrain (s_1 and s_2)
- The length l_1 of the first segment (on the road)

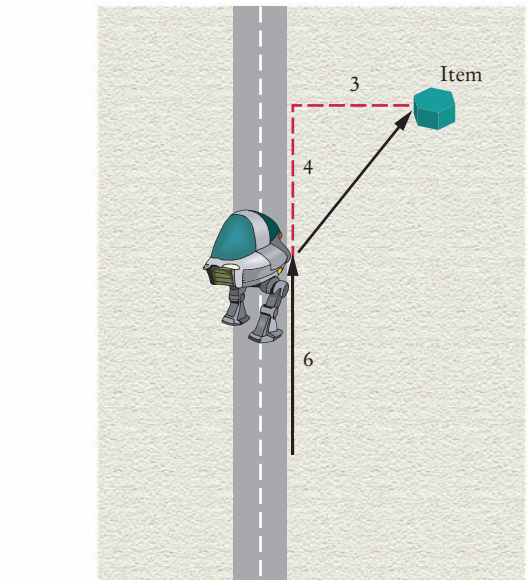


To make the problem more concrete, let's assume the following dimensions:

The total time is the time for traversing both segments. The time to traverse the first segment is simply the length of the segment divided by the speed: 6 km divided by 5 km/h, or 1.2 hours.



To compute the time for the second segment, we first need to know its length. It is the hypotenuse of a right triangle with side lengths 3 and 4.



Therefore, its length is $\sqrt{3^2 + 4^2} = 5$. At 2 km/h, it takes 2.5 hours to traverse it. That makes the total travel time 3.7 hours.

This computation gives us enough information to devise an algorithm for the total travel time with arbitrary parameters.

Time for segment 1 = l_1 / s_1
Length of segment 2 = square root of dx^2 + $(dy - l_1)^2$
Time for segment 2 = l_2 / s_2
Total time = time for segment 1 + time for segment 2

Translated into C++, the computations are

```
double segment1_time = segment1_length / segment1_speed;
double segment2_length = sqrt(pow(x_distance, 2)
   + pow(y_distance - distance_on_road, 2));
double segment2_time = segment2_length / segment2_speed;
double total_time = segment1_time + segment2_time;
```

Note that we use variable names that are longer and more descriptive than *dx* or s_1. When you do hand calculations, it is convenient to use the shorter names, but you should change them to descriptive names in your program.