Greedy

Lecture Flow

- Pre-requisites
- Warm up
- Motivating Problem
- Greedy And Terms
- Properties of Greedy
- Solve Problems
- Proving Correctness
- Pitfalls
- Quote of the day

Prerequisites

- Arrays

Warm up

Feasible Solution

- A feasible solution is a solution that satisfies all the constraints of a given problem. In other words, it is a solution that meets all the specified requirements or conditions without violating any restrictions.

Feasible Solution - Example

You have a set of non-negative integers, and the goal is to find a subset of these integers such that the sum of the subset is equal to a given target value.

Example:
Integers: {3, 1, 4, 6, 2}, target = 5

Feasible Solutions:
{1, 4}, {2, 3}

Optimal Solution

- An optimal solution is the best possible solution to a given problem, according to a specific criterion.

Optimal Solution - Example

Problem:
Given an array of numbers, find the maximum element in the array.

Example:
Consider the array: [4, 7, 1, 9, 3]

Optimal Solution: 9

Optimization Problem

- An optimization problem is a type of problem where the goal is to find the best possible solution from a set of feasible solutions.
- You typically aim to either maximize or minimize a certain task.
- A simple problem would be finding the maximum number in an array.

Common components of an optimization problem

1. Objective Function
2. Decision Variables
3. Constraints

1. Objective Function

The function that needs to be either maximized or minimized.

Purpose:

Represents the goal or outcome that you want to achieve.

Example:

Maximize profit, minimize cost, maximize efficiency.

2. Decision Variables

The variables that you can adjust or control to influence the objective function.

Purpose:
Represents the parameters or choices that you need to make.

Example:
Quantity of products to produce, allocation of resources, time allocation.

3. Constraints

Restrictions or limitations on the values that the decision variables can take.

Purpose:
Ensures that solutions are feasible and realistic.

Example:
Budget constraints, time constraints, resource constraints.

Can you think of any real life optimization problems?

Real life Optimization Problems

1. Grocery Shopping

Problem: You want to buy groceries with a limited budget and maximize the number of items you can purchase.

Objective Function: Maximize the number of grocery items.

Decision Variables: Quantities of different items to buy.

Constraints: Budget constraint.

Real life Optimization Problems

2. Study Time Allocation

Problem: You have limited time to study for exams and want to maximize your overall grades.

Objective Function: Maximize overall grades.

Decision Variables: Time allocated to each subject.

Constraints: Time constraint.

Motivating Problem

What are the components of the motivating problem?

Problem: ?

Objective Function: ?

Decision Variables: ?

Constraints: ?

What are the components of the motivating problem?

Problem: Given a bag containing items with numbers 1, 0, and -1. Maximize the sum of the numbers written on the selected items.

Objective Function: Maximize sum of the numbers

Decision Variables: play around with 0, 1, -1

Constraints: Select exactly k items

Mathematical Representation

It is a good practice to represent the components with numerical values

X_i = item
$number_i$ = number written on the ith item

Objective function: maximize $Z = \Sigma^{n}_{i=1} number_i$

Decision Variables: A piecewise function with -1,0,1

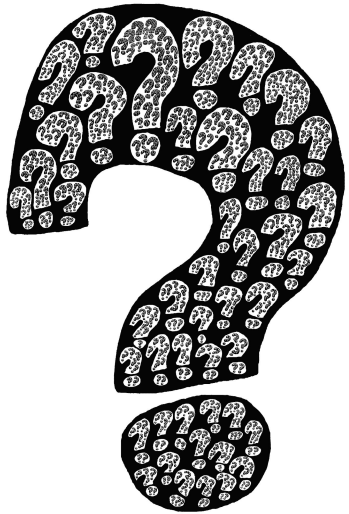
Constraints: $\Sigma^{n}_{i=1} x_i = k$

Greedy

Greedy is a method for solving optimization problems.

But How?

How do you think the greedy method solves an optimization problem just by it's name?



Greedy

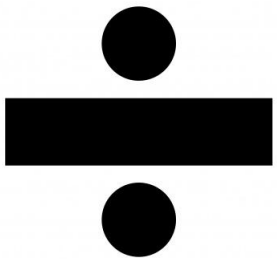
The greedy method is an approach to solving problems by making locally optimal choices at each stage with the hope of finding a global optimum.

It follows the idea that, at each step, the approach selects the best available option without considering the consequences of that choice in future steps.

Terms

Subproblem

A subproblem is a smaller instance of a larger problem that shares some form of similarity.



Subproblem - Example

Problem:
Calculate the factorial of a non-negative integer n, denoted as $n!$.

Subproblem:
The problem of calculating $n!$ can be broken down into a subproblem of calculating $(n-1)!$ and continues downward to more subproblems.

Local Optimum

A local optimum is a choice that is best among a specific set of options.

Global Optimum

A global optimum, also known as the global minimum or maximum, refers to the best possible solution of an optimization problem across the entire feasible solution space.

Global and local Optimum - Example

Problem

Imagine you are trying to optimize the temperature in a room using a simple thermostat. Your goal is to achieve a comfortable temperature.

Global and local Optimum - Example

Local Optimum

If you set the thermostat to a certain temperature and wait for the room to stabilize, you may reach a point where the temperature is relatively stable for now.

But is that temperature always going to be comfortable?

What if it rains or it becomes super hot outside.

Global and local Optimum - Example

Global Optimum

There might be a better temperature setting that would be more comfortable for a broader range of conditions, such as different weather or occupancy patterns.

Properties of Greedy

1. Greedy Choice Property: A greedy algorithm makes the locally optimal choice at each step without considering the overall structure of the problem.

2. Optimal Substructure: The problem should have the property that an optimal solution to the entire problem can be constructed from optimal solutions to its subproblems.

Selection Rule

In greedy, the selection rule is a key decision-making principle that guides the algorithm to choose the next best option at each step.

Heuristic

A heuristic can be considered as a type of selection rule, particularly in the context of optimization algorithms.

A heuristic is a practical and often simplified rule or strategy that guides decision-making

Greedy Example With A Heuristic

Let's take the motivating problem components

Problem: Given a bag containing items with numbers 1, 0, and -1. Maximize the sum of the numbers written on the selected items.

Objective Function: Maximize sum of the numbers

Decision Variables: Play around with 0, 1, -1

Constraints: Select exactly k items

Let's Think Greedy Now

What are we being greedy on?

What heuristic will enable us to fulfill that greed?

What are we being greedy on?

The numbers we select.

What heuristic will enable us to fulfill that greed?

Prioritizes selecting positive values to maximize the sum.

Be careful what you are greedy on

Problem:

Given a set of coin denominations (e.g., 1, 5, 10 cents) and a target amount, find the minimum number of coins needed to make the target amount.

Be careful what you are greedy on

Problem:

Given a set of coin denominations (e.g., 1, 5, 10 cents) and a target amount, find the minimum number of coins needed to make the target amount.

This fails with greedy!

Termination Condition

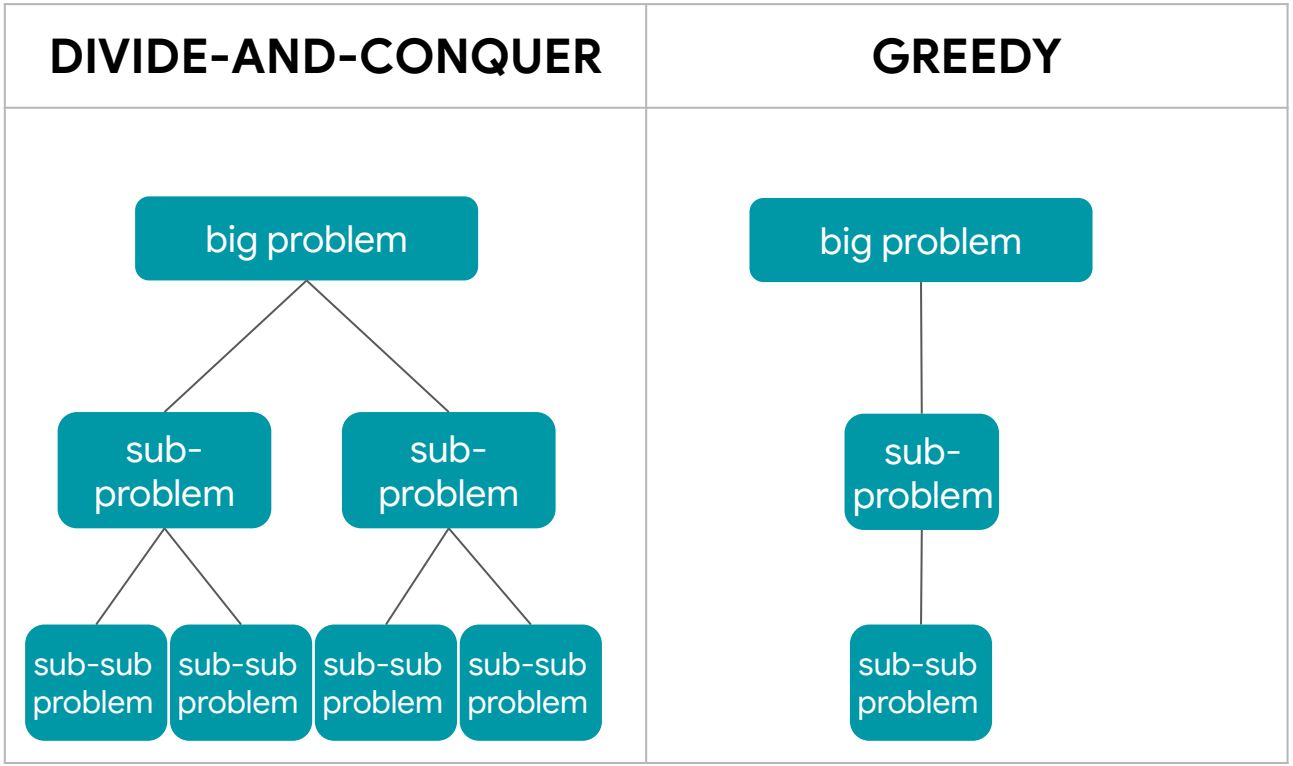
In greedy, the termination condition is typically determined by the nature of the problem being solved. The algorithm continues making locally optimal choices until a certain condition is met.

Greedy vs Divide-and-Conquer

Greedy algorithms make locally optimal choices at each step with the hope that they lead to a globally optimal solution.

Divide and conquer algorithms break down a problem into smaller subproblems, solve them independently, and combine their solutions to obtain the final result.

Greedy vs Divide-and-Conquer



Solve This

Greedy Bootcamp (To be done in lecture with instructor)

1

2

3

4

5

Africa To Silicon Valley

Once we know the right greedy choice, it's easy to implement it.

But how can we know if a greedy choice is right?

Proving Correctness

Proofing or analyzing the correctness and optimality of greedy algorithms involves demonstrating that the algorithm consistently makes locally optimal choices at each step, leading to a globally optimal solution.

Proofing Mechanisms

1. Proof by Induction
2. Proof by Contradiction
3. Exchange Argument

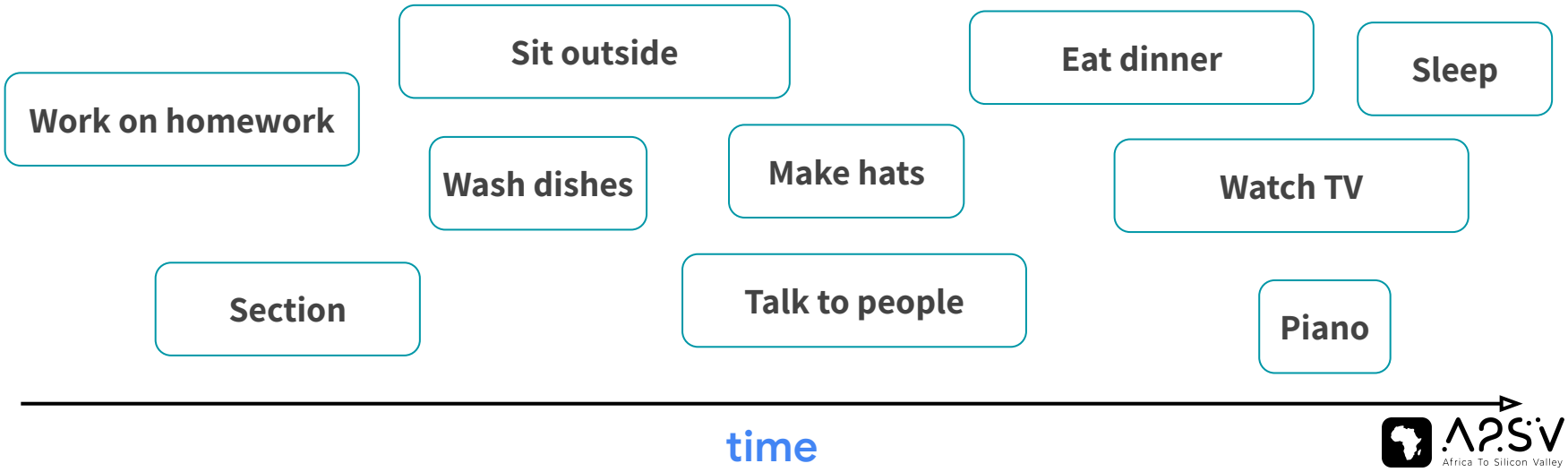
Proof by Induction

Proof by Induction

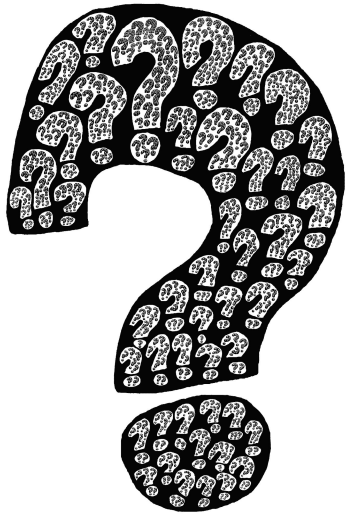
The technique involves proving a statement for a base case and then demonstrating that if the statement holds for any given case, it must also hold for the next case.

ACTIVITY SELECTION PROBLEM

Problem: n activities with start times and finish times
Constraint: All activities are equally important, but you can only do 1 activity at a time!
Objective function: Maximize the number of activities you can do



How would you solve this question?



Africa To Silicon Valley

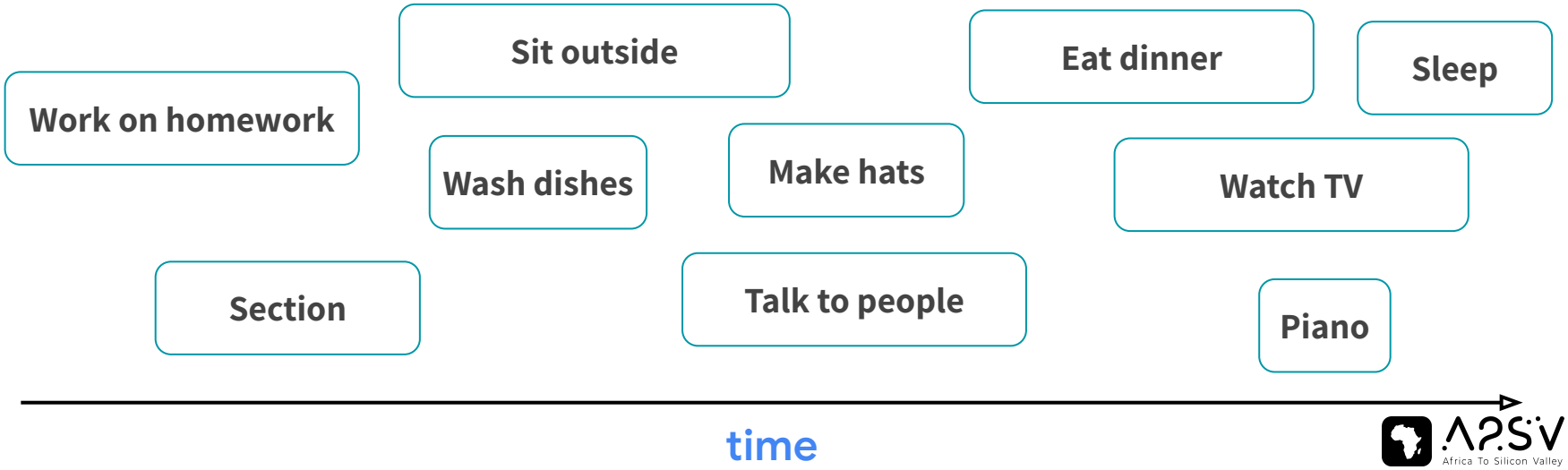
Activity Selection : Options

In what order should you greedily add activities? Here are 3 ideas:

1) Be impulsive: choose activities in ascending order of start times

2) Avoid commitment: choose activities in ascending order of length

3) Finish fast: choose activities in ascending order of end times



Activity Selection : Options

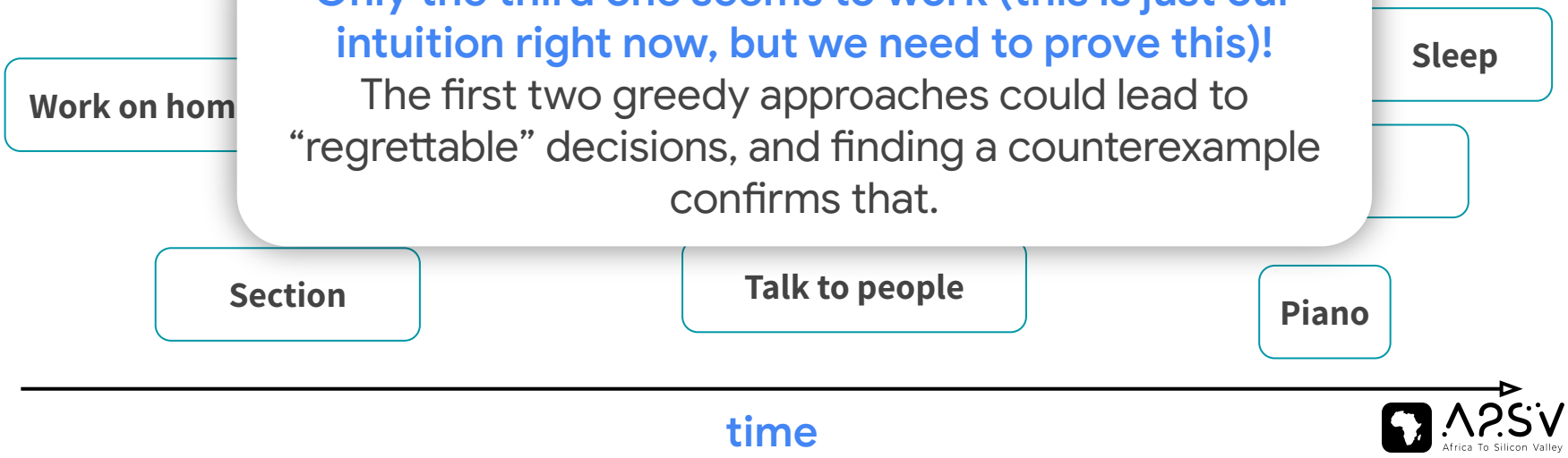
In what order should you greedily add activities? Here are 3 ideas:

1) Be impulsive: choose activities in ascending order of start times

2) Avoid commitment: choose activities in ascending order of length

3) Finish fast: choose activities in ascending order of end times

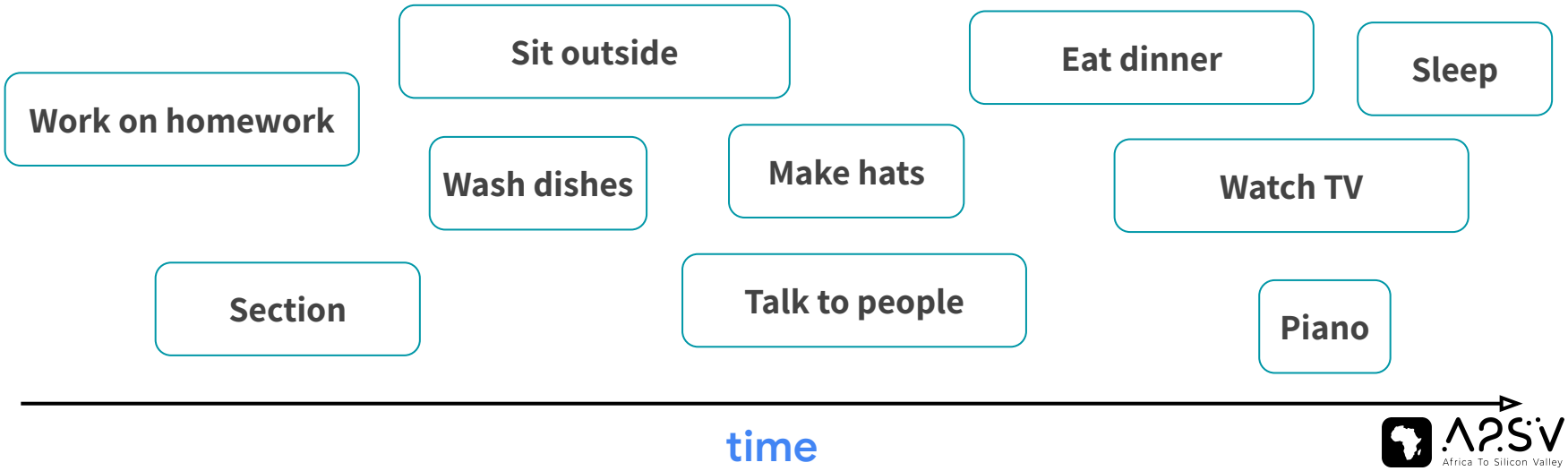
Only the third one seems to work (this is just our intuition right now, but we need to prove this)!
The first two greedy approaches could lead to "regrettable" decisions, and finding a counterexample confirms that.



AASV
Africa To Silicon Valley

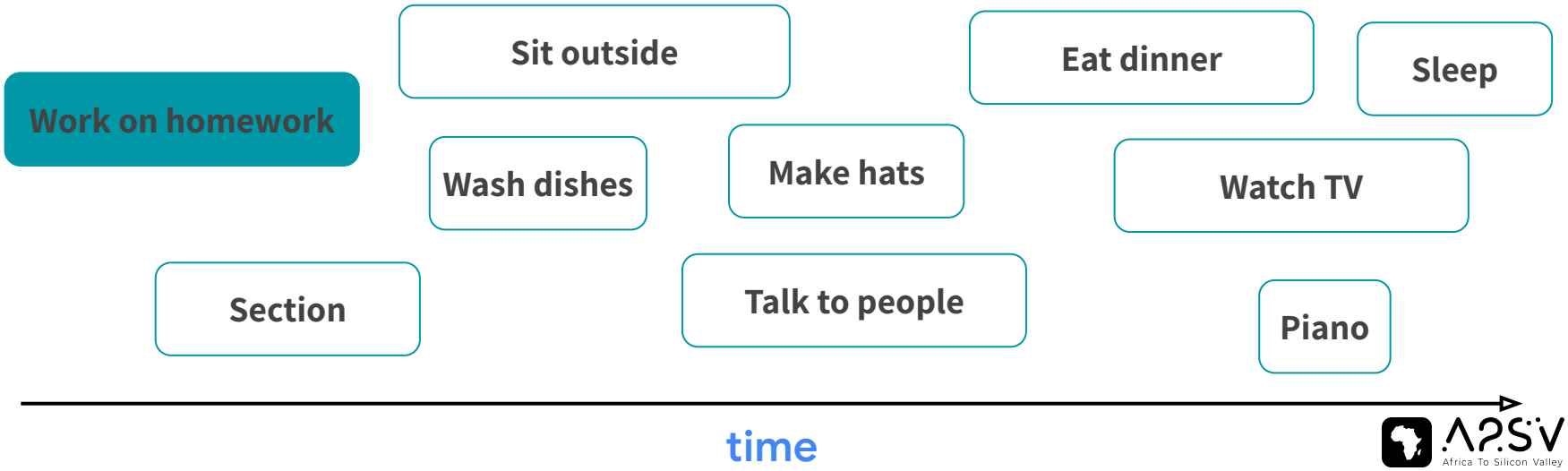
Our Greed Algorithm

Pick an available activity with the smallest finish time & repeat.



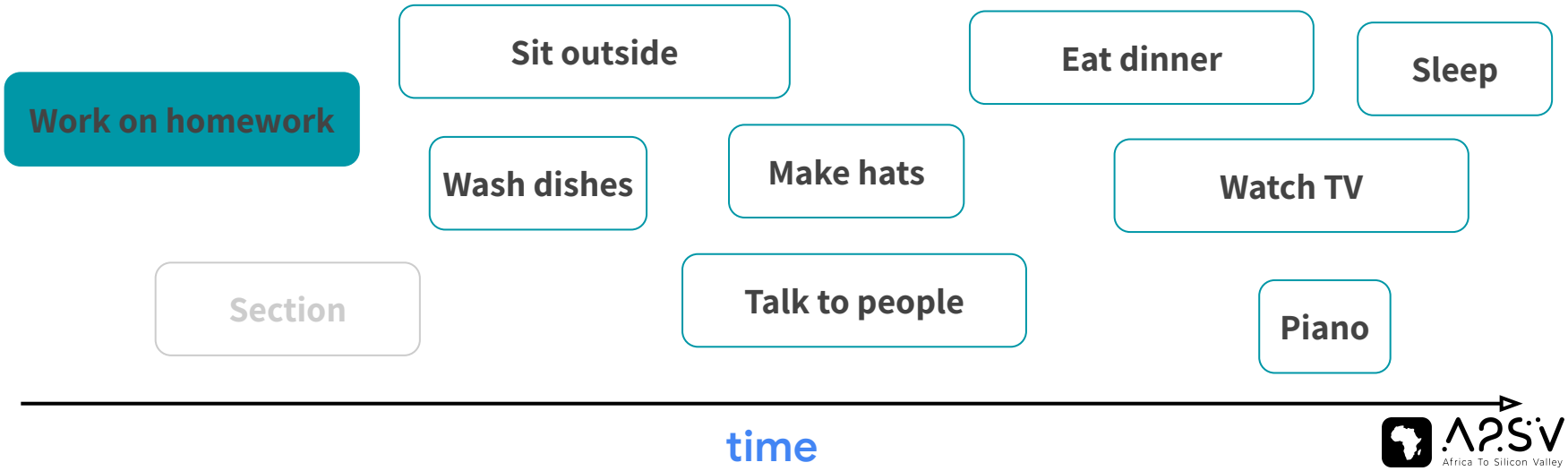
Our Greed Algorithm

Pick an available activity with the smallest finish time & repeat.



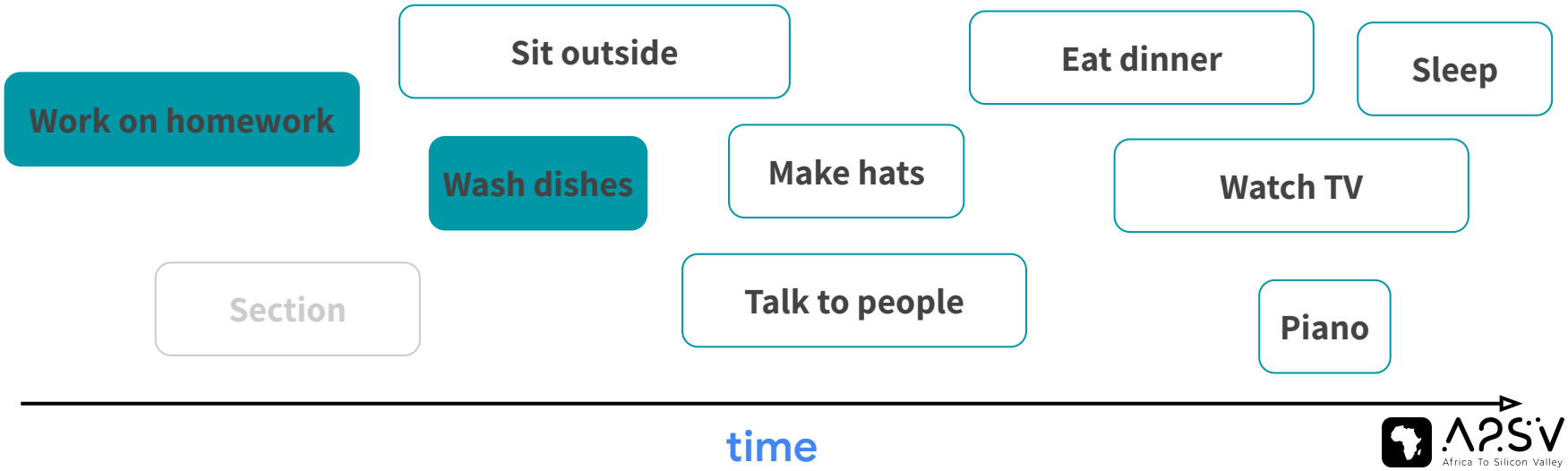
Our Greed Algorithm

Pick an available activity with the smallest finish time & repeat.



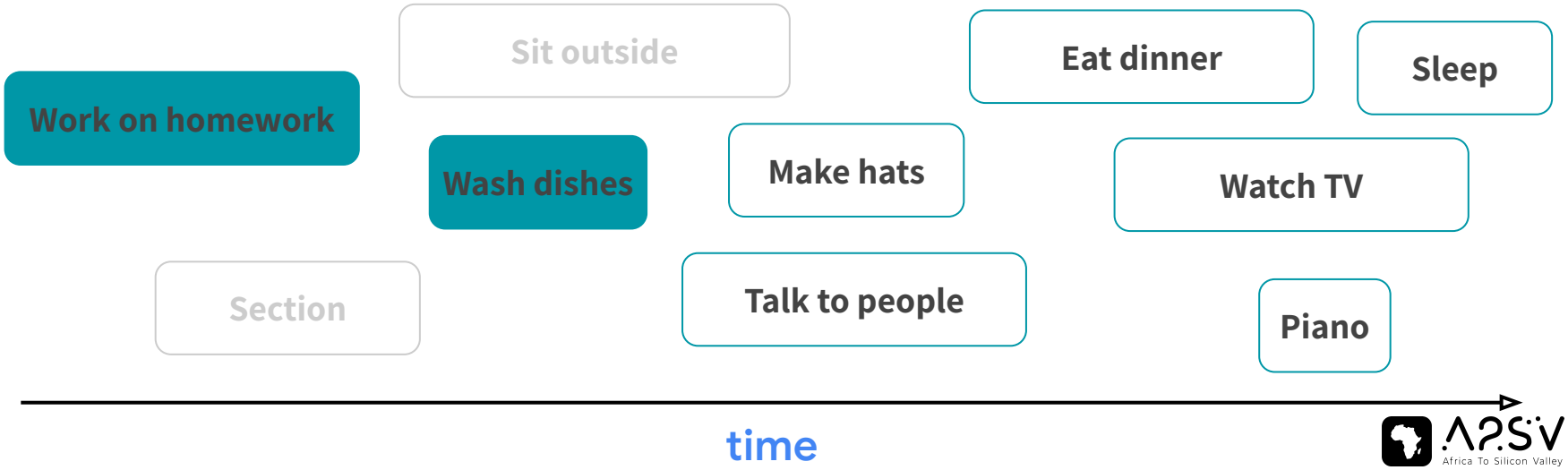
Our Greed Algorithm

Pick an available activity with the smallest finish time & repeat.



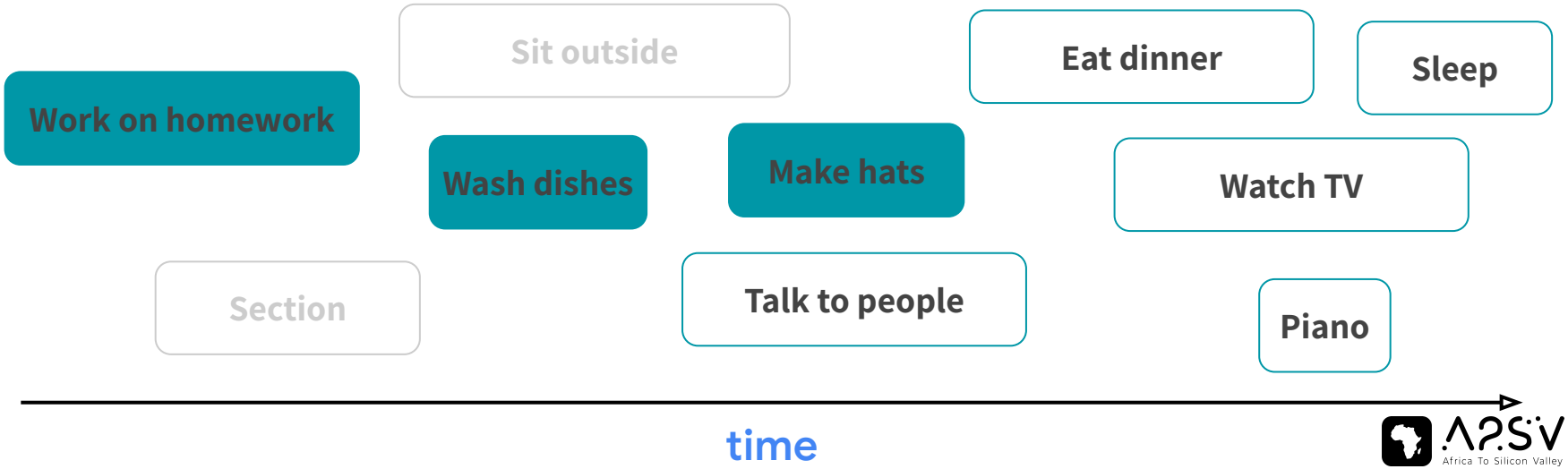
Our Greed Algorithm

Pick an available activity with the smallest finish time & repeat.



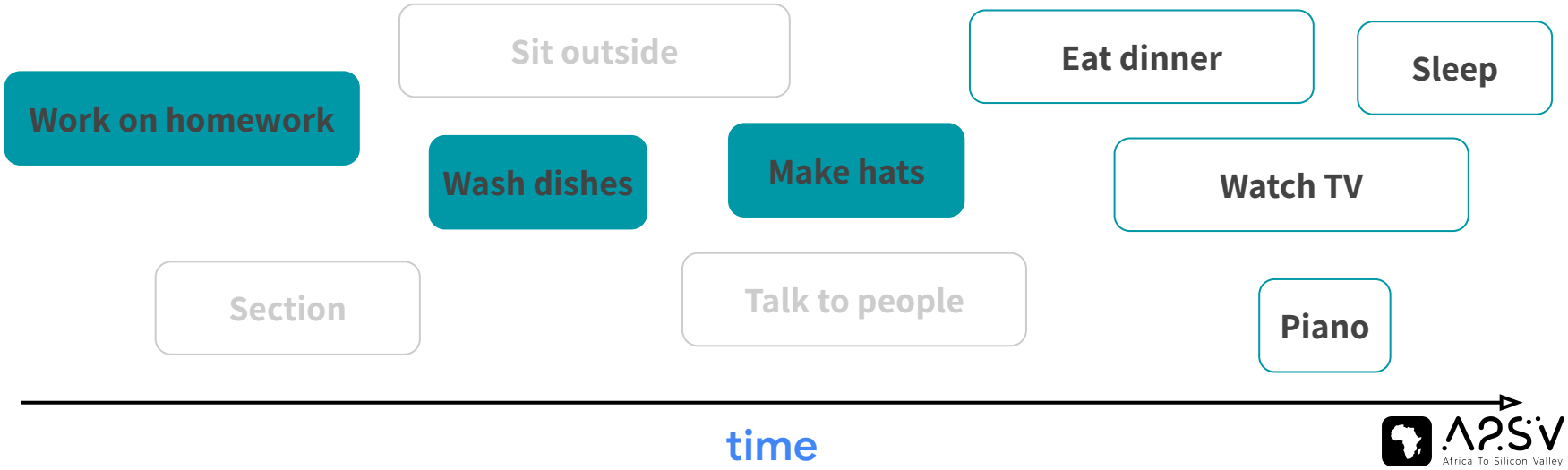
Our Greed Algorithm

Pick an available activity with the smallest finish time & repeat.



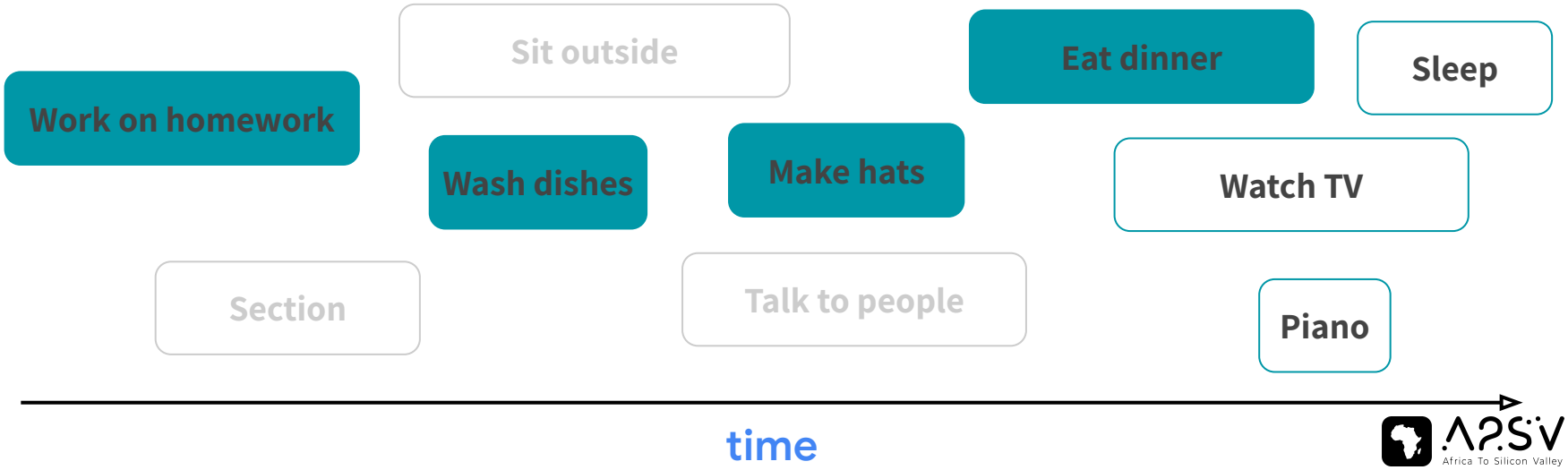
Our Greed Algorithm

Pick an available activity with the smallest finish time & repeat.



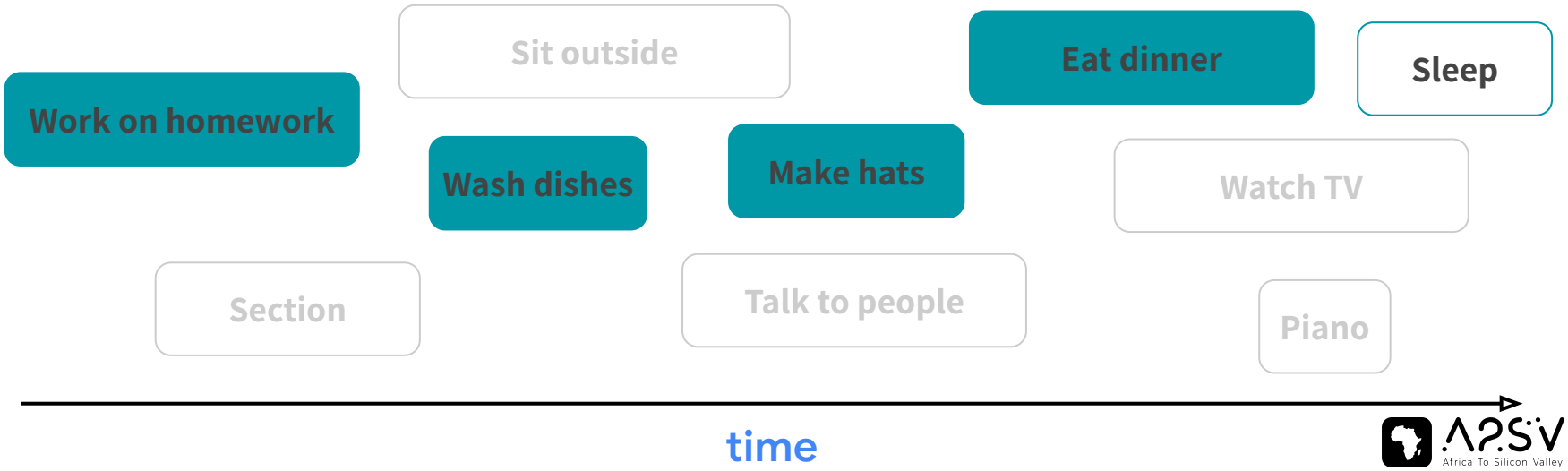
Our Greed Algorithm

Pick an available activity with the smallest finish time & repeat.



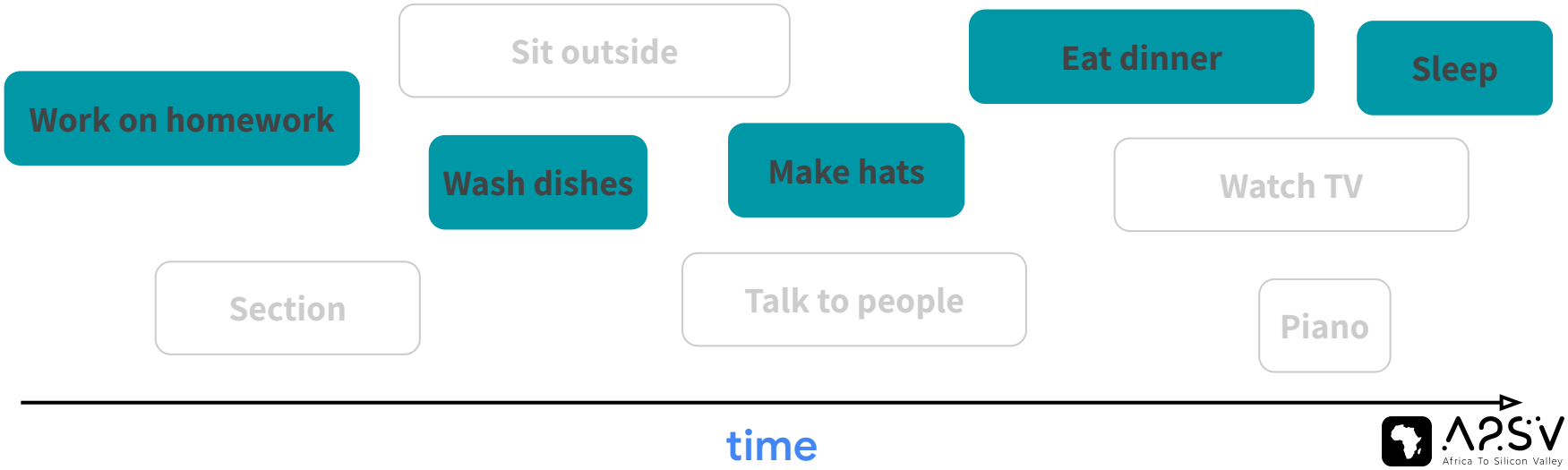
Our Greed Algorithm

Pick an available activity with the smallest finish time & repeat.



Our Greed Algorithm

Pick an available activity with the smallest finish time & repeat.



Why is it Greedy?

At each step in the algorithm, we make a choice (pick the available activity with the smallest finish time) and never look back.

How do we know that this greedy algorithm is correct?

(Proving correctness is the hard part!)

To prove correctness, we use induction.

In induction, we have:

- Base case
  - The initial case can be part of an optimal solution
- Inductive step
  - If the past K steps are part of the optimal solution and we make the k+1 choice greedily, we can still reach an optimal solution.

Activity Selection : Correctness

BASE CASE

After adding 0 activities, there is an optimal solution we can reach from that (any solution extends 0 activities).

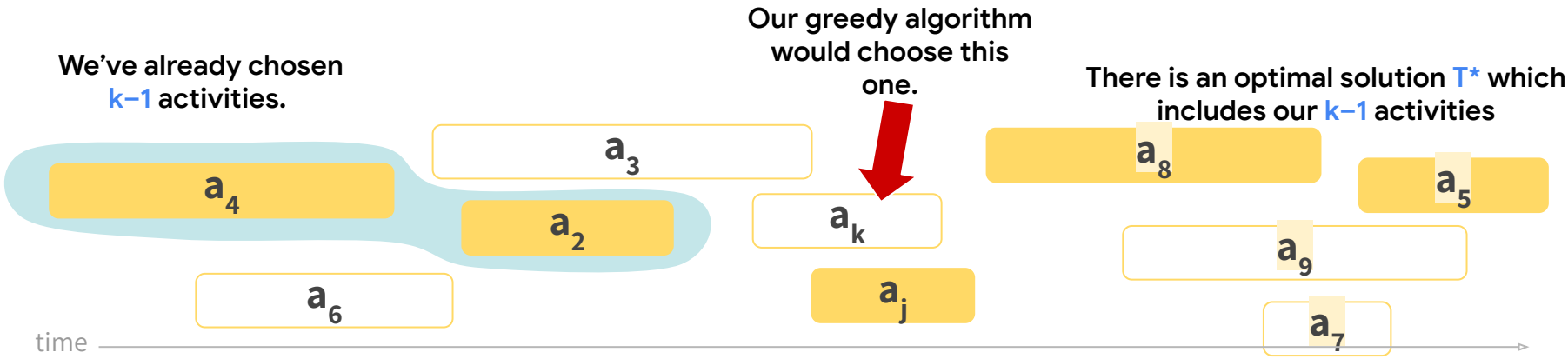
INDUCTIVE STEP *(weak induction)*

Suppose we've already chosen (k−1) activities, and we can reach an optimal solution from these choices. After adding the k^{th} activity, we'll show that we can still reach an optimal solution from these k activities. Let's do this step!

CONCLUSION

After adding the last activity, we can reach an optimal solution from the current solution. Since there is no remaining activity that we could still fit in, the current solution is optimal.

To prove the inductive step, we can consider the case where there's an optimal solution that doesn't include our greedy choice.



Our greedy choice of a_k doesn't belong in T*.

Then, let a_j be the activity in T* after a_{k-1}.

Since a_k was chosen because it has the smallest end time, we know it ends before a_j.

Thus, a_k doesn't conflict with anything in T* after a_j. Swapping a_k with a_j would preserve optimality!

Pitfalls and Weaknesses

1. Greedy-Choice Property does not always lead to the Global Optimum (Our intuition is wrong)
2. Failure to Consider Future Consequences
3. Dependency on Sorting
4. Proof Challenges (The biggest weakness)

Practice Problems

Lemonade Change

Minimum Number of Arrows to Burst Balloons

Rabbits in Forest

Minimum Moves to Reach Target Score

Quote of the Day

"A problem well stated is a problem half solved."

— John Dewey