

COMP9414 Artificial Intelligence

Assignment 1: Constraint Satisfaction Search

@Authors: **Wayne Wobcke, Alfred Krzywicki, Stefano Mezza**

Due Date: Week 5, Friday, October 17, 5.00pm

Objective

This assignment concerns developing optimal solutions to a scheduling problem inspired by the scenario of a manufacturing plant that has to fulfil multiple customer orders with varying deadlines, but where there may be constraints on tasks and on relationships between tasks. Any number of tasks can be scheduled at the same time, but it is possible that some tasks cannot be finished before their deadline. A task finishing late is acceptable, however incurs a cost, which for this assignment is a simple (dollar) amount per hour that the task is late.

A *fuzzy scheduling* problem in this scenario is simplified by ignoring customer orders and having just one machine and a number of *tasks*, each with a fixed duration in hours. Each task must start and finish on the same day, within working hours (9am to 5pm). In addition, there can be *constraints*, both on single tasks and between two tasks. One type of constraint is that a task can have a deadline, which can be “hard” (the deadline must be met in any valid schedule) or “soft” (the task may be finished late – though still at or before 5pm – but with a “cost” per hour for missing the deadline). The aim is to develop an overall schedule for all the tasks (in a single week) that minimizes the total cost of all the tasks that finish late, provided that all the hard constraints on tasks are satisfied.

More technically, this assignment is an example of a *constraint optimization problem* (or *constrained optimization problem*), a problem that has constraints like a standard Constraint Satisfaction Problem (CSP), but also a *cost* associated with each solution. For this assignment, we will use a *greedy* algorithm to find optimal solutions to fuzzy scheduling problems that are specified as text strings. However, unlike the greedy search algorithm described in the lectures on search, this greedy algorithm has the property that it is guaranteed to find an optimal solution for any problem (if a solution exists).

The assignment will use the AI^{Python} code of Poole & Mackworth. You are given code to translate fuzzy scheduling problems specified as text strings into CSPs with a cost, and you are given code for several constraint solving algorithms – based on domain splitting and arc consistency, and based on depth-first search. The assignment will be to implement some missing procedures and to analyse the performance of the constraint solving methods, both analytically and experimentally.

Submission Instructions

- This is an individual assignment.
- Write your answers in **this** notebook and submit **this** notebook on Moodle under **Assignment 1**.
- Name your submission `<zid>-<firstname>-<lastname>.ipynb` where `<firstname>-<lastname>` is your **real** (not Moodle) name.
- Make sure you set up AI Python (as done below) so the code can be run on either CSE machines or a marker's own machine.
- Do not submit any AI Python code. Hence do not change any AI Python code to make your code run.
- Make sure your notebook runs cleanly (restart the kernel, clear all outputs and run each cell to check).
- After checking that your notebook runs cleanly, run all cells and submit the notebook **with** the outputs included (do not submit the empty version).
- Make sure images (for plots/graphs) are **included** in the notebook you submit (sometimes images are saved on your machine but are not in the notebook).
- Do not modify the existing code in this notebook except to answer the questions. Marks will be given as and where indicated.
- If you want to submit additional code (e.g. for generating plots), add that at the end of the notebook.
- **Important: Do not distribute any of this code on the Internet. This includes ChatGPT. Do not put this assignment into any LLM.**

Late Penalties

Standard UNSW late penalties apply (5% of the value of the assignment per day or part day late).

Note: Unlike the CSE systems, there is no grace period on Moodle. The due date and time is 5pm **precisely** on Friday October 17.

Important: You can submit as many times as you want before the due date, but if you do submit before the due date, you cannot submit on Moodle after the due date. If you do not submit before the due date, you can submit on Moodle after the due date.

Plagiarism

Remember that ALL work submitted for this assignment must be your own work and no sharing or copying of code or answers is allowed. You may discuss the assignment with other students but must not collaborate on developing answers to the questions. You

may use code from the Internet only with suitable attribution of the source. You may not use ChatGPT or any similar software to generate any part of your explanations, evaluations or code. Do not use public code repositories on sites such as github or file sharing sites such as Google Drive to save any part of your work – make sure your code repository or cloud storage is private and do not share any links. This also applies after you have finished the course, as we do not want next year's students accessing your solution, and plagiarism penalties can still apply after the course has finished.

All submitted assignments will be run through plagiarism detection software to detect similarities to other submissions, including from past years. You should **carefully** read the UNSW policy on academic integrity and plagiarism (linked from the course web page), noting, in particular, that collusion (working together on an assignment, or sharing parts of assignment solutions) is a form of plagiarism.

Finally, do not use any contract cheating “academies” or online “tutoring” services. This counts as serious misconduct with heavy penalties up to automatic failure of the course with 0 marks, and expulsion from the university for repeat offenders.

Fuzzy Scheduling

A CSP for this assignment is a set of variables representing tasks, binary constraints on pairs of tasks, and unary constraints (hard or soft) on tasks. The domains are all the working hours in one week, and a task duration is in hours. Days are represented (in the input and output) as strings ‘mon’, ‘tue’, ‘wed’, ‘thu’ and ‘fri’, and times are represented as strings ‘9am’, ‘10am’, ‘11am’, ‘12pm’, ‘1pm’, ‘2pm’, ‘3pm’, ‘4pm’ and ‘5pm’. The only possible values for the start and end times of a task are combinations of a day and times, e.g. ‘mon 9am’. Each task name is a string (with no spaces), and the only soft constraints are the soft deadline constraints.

There are three types of constraint:

- **Binary Constraints:** These specify a hard requirement for the relationship between two tasks.
- **Hard Domain Constraints:** These specify hard requirements for the tasks themselves.
- **Soft Deadline Constraints:** These constraints specify that a task may finish late, but with a given cost.

Each soft constraint has a function defining the *cost* associated with violating the preference, that the constraint solver must minimize, while respecting all the hard constraints. The *cost* of a solution is simply the sum of the costs for the soft constraints that the solution violates (and is always a non-negative integer).

This is the list of possible constraints for a fuzzy scheduling problem (comments below are for explanation and do **not** appear in the input specification; however, the code we supply *should* work with comments that take up a full line):

```

# binary constraints
constraint, <t1> before <t2>                      # t1 ends when or before
t2 starts
constraint, <t1> after <t2>                         # t1 starts after or when
t2 ends
constraint, <t1> same-day <t2>                      # t1 and t2 are scheduled
on the same day
constraint, <t1> starts-at <t2>                     # t1 starts exactly when
t2 ends

# hard domain constraints
domain, <t>, <day>, hard                           # t
starts on given day at any time
domain, <t>, <time>, hard                            # t
starts at given time on any day
domain, <t>, starts-before <day> <time>, hard      # t
starts at or before day, time
domain, <t>, starts-after <day> <time>, hard        # t
starts at or after day, time
domain, <t>, ends-before <day> <time>, hard         # t
ends at or before day, time
domain, <t>, ends-after <day> <time>, hard          # t
starts at or after day, time
domain, <t>, starts-in <day1> <time1>-<day2> <time2>, hard # day-
time range for start time; includes day1, time1 and day2, time2
domain, <t>, ends-in <day1> <time1>-<day2> <time2>, hard   # day-
time range for end time; includes day1, time1 and day2, time2
domain, <t>, starts-before <time>, hard              # t
starts at or before time on any day
domain, <t>, ends-before <time>, hard                # t
ends at or before time on any day
domain, <t>, starts-after <time>, hard              # t
starts at or after time on any day
domain, <t>, ends-after <time>, hard                # t
ends at or after time on any day

# soft deadline constraint
domain, <t>, ends-by <day> <time> <cost>, soft       # cost per
hour of missing deadline

```

The input specification will consist of several “blocks”, listing the tasks, binary constraints, hard unary constraints and soft deadline constraints for the given problem. A “declaration” of each task will be included before it is used in a constraint. A sample input specification is as follows. Comments are for explanation and do **not** have to be included in the input.

```

# two tasks with two binary constraints and soft deadlines
task, t1 3
task, t2 4
# two binary constraints
constraint, t1 before t2
constraint, t1 same-day t2
# domain constraint
domain, t2 mon

```

```
# soft deadline constraints
domain, t1 ends-by mon 3pm 10
domain, t2 ends-by mon 3pm 10
```

Preparation

1. Set up AlPython

You will need AlPython for this assignment. To find the aipython files, the aipython directory has to be added to the Python path.

Do this temporarily, as done here, so we can find AlPython and run your code (you will not submit any AlPython code).

You can add either the full path (using `os.path.abspath()`), or as in the code below, the relative path.

```
In [1]: import sys
sys.path.append('aipython') # change to your directory
sys.path # check that aipython is now on the path
```

```
Out[1]: ['d:\\vsc\\python\\python313.zip',
'd:\\vsc\\python\\DLLs',
'd:\\vsc\\python\\Lib',
'd:\\vsc\\python',
 '',
'd:\\vsc\\python\\Lib\\site-packages',
'd:\\vsc\\python\\Lib\\site-packages\\win32',
'd:\\vsc\\python\\Lib\\site-packages\\win32\\lib',
'd:\\vsc\\python\\Lib\\site-packages\\Pythonwin',
'aipython']
```

2. Representation of Day Times

Input and output are day time strings such as 'mon 10am' or a range of day time strings such as 'mon 10am-mon 4pm'.

The CSP will represent these as integer hour numbers in the week, ranging from 0 to 39.

The following code handles the conversion between day time strings and hour numbers.

```
In [2]: # -*- coding: utf-8 -*-

""" day_time string format is a day plus time, e.g. Mon 10am, Tue 4pm, or just T
    if only day or time, returns day number or hour number only
    day_time strings are converted to and from integer hours in the week from 0
"""
class Day_Time():
    num_hours_in_day = 8
    num_days_in_week = 5

    def __init__(self):
        self.day_names = ['mon', 'tue', 'wed', 'thu', 'fri']
        self.time_names = ['9am', '10am', '11am', '12pm', '1pm', '2pm', '3pm', '4pm']
```

```

def string_to_week_hour_number(self, day_time_str):
    """ convert a single day_time into an integer hour in the week """
    value = None
    value_type = None
    day_time_list = day_time_str.split()
    if len(day_time_list) == 1:
        str1 = day_time_list[0].strip()
        if str1 in self.time_names: # this is a time
            value = self.time_names.index(str1)
            value_type = 'hour_number'
        else:
            value = self.day_names.index(str1) # this is a day
            value_type = 'day_number'
        # if not day or time, throw an exception
    else:
        value = self.day_names.index(day_time_list[0].strip())*self.num_hour
        + self.time_names.index(day_time_list[1].strip())
        value_type = 'week_hour_number'
    return (value_type, value)

def string_to_number_set(self, day_time_list_str):
    """ convert a list of day-times or ranges 'Mon 9am, Tue 9am-Tue 4pm' int
        e.g. 'mon 9am-1pm, mon 4pm' -> [0,1,2,3,4,7]
    """
    number_set = set()
    type1 = None
    for str1 in day_time_list_str.lower().split(','):
        if str1.find('-') > 0:
            # day time range
            type1, v1 = self.string_to_week_hour_number(str1.split('-')[0].s
            type2, v2 = self.string_to_week_hour_number(str1.split('-')[1].s
            if type1 != type2: return None # error, types in range spec are
            number_set.update({n for n in range(v1, v2+1)})
        else:
            # single day time
            type2, value2 = self.string_to_week_hour_number(str1)
            if type1 != None and type1 != type2: return None # error: type i
            type1 = type2
            number_set.update({value2})
    return (type1, number_set)

# convert integer hour in week to day time string
def week_hour_number_to_day_time(self, week_hour_number):
    hour = self.day_hour_number(week_hour_number)
    day = self.day_number(week_hour_number)
    return self.day_names[day] + ' ' + self.time_names[hour]

# convert integer hour in week to integer day and integer time in day
def hour_day_split(self, week_hour_number):
    return (self.day_hour_number(week_hour_number), self.day_number(week_hou

# convert integer hour in week to integer day in week
def day_number(self, week_hour_number):
    return int(week_hour_number / self.num_hours_in_day)

# convert integer hour in week to integer time in day
def day_hour_number(self, week_hour_number):
    return week_hour_number % self.num_hours_in_day

```

```

def __repr__(self):
    day_hour_number = self.week_hour_number % self.num_hours_in_day
    day_number = int(self.week_hour_number / self.num_hours_in_day)
    return self.day_names[day_number] + ' ' + self.time_names[day_hour_number]

```

3. Constraint Satisfaction Problems with Costs over Tasks with Durations

Since AI Python does not provide the CSP class with an explicit cost, we implement our own class that extends `CSP`.

We also store the cost functions and the durations of all tasks explicitly in the CSP.

The durations of the tasks are used in the `hold` function to evaluate constraints.

```

In [3]: from cspProblem import CSP, Constraint

# We need to override Constraint, because tasks have durations
class Task_Constraint(Constraint):
    """A Task_Constraint consists of
    * scope: a tuple of variables
    * spec: text description of the constraint used in debugging
    * condition: a function that can applied to a tuple of values for the variables
    * durations: durations of all tasks
    * func_key: index to the function used to evaluate the constraint
    """
    def __init__(self, scope, spec, condition, durations, func_key):
        super().__init__(scope, condition, spec)
        self.scope = scope
        self.condition = condition
        self.durations = durations
        self.func_key = func_key

    def holds(self, assignment):
        """returns the value of Constraint con evaluated in assignment.

        precondition: all variables are assigned in assignment

        CSP has only binary constraints
        condition is in the form week_hour_number1, week_hour_number2
        add task durations as appropriate to evaluate condition
        """
        if self.func_key == 'before':
            # t1 ends before t2 starts, so we need add duration to t1 assignment
            ass0 = assignment[self.scope[0]] + self.durations[self.scope[0]]
            ass1 = assignment[self.scope[1]]
        elif self.func_key == 'after':
            # t2 ends before t1 starts so we need add duration to t2 assignment
            ass0 = assignment[self.scope[0]]
            ass1 = assignment[self.scope[1]] + self.durations[self.scope[1]]
        elif self.func_key == 'starts-at':
            # t1 starts exactly when t2 ends, so we need add duration to t2 assignment
            ass0 = assignment[self.scope[0]]
            ass1 = assignment[self.scope[1]] + self.durations[self.scope[1]]
        else:
            return self.condition(*tuple(assignment[v] for v in self.scope))
        # condition here comes from get_binary_constraint
        return self.condition(*tuple([ass0, ass1]))

```

```

# implement nodes as CSP problems with cost functions
class CSP_with_Cost(CSP):
    """ cost_functions maps a CSP var, here a task name, to a list of functions """
    def __init__(self, domains, durations, constraints, cost_functions, soft_day_time):
        self.domains = domains
        self.variables = self.domains.keys()
        super().__init__("title of csp", self.variables, constraints)
        self.durations = durations
        self.cost_functions = cost_functions
        self.soft_day_time = soft_day_time
        self.soft_costs = soft_costs
        self.cost = self.calculate_cost()

    # specific to fuzzy scheduling CSP problems
    def calculate_cost(self, context=None, use_lower_bound=True):# context: part

        import math
        if context is None:
            context = {}

        total_cost = 0

        # Iterate over all tasks and their current domains
        for task, domain in self.domains.items():
            # If any task's domain has been wiped out by arc consistency (AC),
            # this node is impossible, so return infinity cost.
            if not domain:
                return math.inf

            # Retrieve the cost function associated with this task.
            funcs = self.cost_functions.get(task)
            if not funcs:
                # Safety fallback: if no cost function exists for this task,
                # treat its cost as 0 to avoid errors.
                funcs = [lambda *args, **kwargs: 0]

            duration = self.durations[task]
            deadline = self.soft_day_time[task]
            soft_cost = int(self.soft_costs[task])

            if task in context:
                # Assigned task: compute the real cost using the cost function
                start = context[task]
                total_cost += sum(
                    f(start, deadline, duration, soft_cost) for f in funcs
                )
            else:
                # Unassigned task
                if use_lower_bound:
                    # 3) Use optimistic lower bound:
                    # take the minimum possible soft cost over all current do
                    best = min(
                        sum(f(st, deadline, duration, soft_cost) for f in funcs)
                        for st in domain
                    )
                    total_cost += best
                # if Lower bound is not requested, skip unassigned tasks

        return total_cost

```

```

def __repr__(self):
    """ string representation of an arc"""
    return "CSP_with_Cost("+str(list(self.domains.keys())))+':'+str(self.cost)

```

This formulates a solver for a CSP with cost as a search problem, using domain splitting with arc consistency to define the successors of a node.

```

In [4]: from cspConsistency import Con_solver, select, partition_domain
from searchProblem import Arc, Search_problem
from operator import eq, le, ge

# rewrites rather than extends Search_with_AC_from_CSP
class Search_with_AC_from_Cost(Search_problem):
    """ A search problem with domain splitting and arc consistency """
    def __init__(self, csp):
        self.cons = Con_solver(csp) # copy of the CSP with access to arc consist
        self.domains = self.cons.make_arc_consistent(csp.domains)
        self.constraints = csp.constraints
        self.cost_functions = csp.cost_functions
        self.durations = csp.durations
        self.soft_day_time = csp.soft_day_time
        self.soft_costs = csp.soft_costs
        csp.domains = self.domains # after arc consistency
        self.csp = csp

    def is_goal(self, node):
        """ node is a goal if all domains have exactly 1 element """
        return all(len(node.domains[var]) == 1 for var in node.domains)

    def start_node(self):
        return CSP_with_Cost(self.domains, self.durations, self.constraints,
                             self.cost_functions, self.soft_day_time, self.soft_)

    def neighbors(self, node):
        """returns the neighboring nodes of node.
        """
        neights = []
        var = select(x for x in node.domains if len(node.domains[x]) > 1) # chooses
        if var:
            dom1, dom2 = partition_domain(node.domains[var])
            self.display(2, "Splitting", var, "into", dom1, "and", dom2)
            to_do = self.cons.new_to_do(var, None)
            for dom in [dom1, dom2]:
                newdoms = node.domains | {var: dom} # overwrite domain of var with
                cons_doms = self.cons.make_arc_consistent(newdoms, to_do)
                if all(len(cons_doms[v]) > 0 for v in cons_doms):
                    # all domains are non-empty
                    # make new CSP_with_Cost node to continue the search
                    csp_node = CSP_with_Cost(cons_doms, self.durations, self.constraints,
                                            self.cost_functions, self.soft_day_time, self.soft_
                    neights.append(Arc(node, csp_node)))
                else:
                    self.display(2, "...", var, "in", dom, "has no solution")
        return neights

    def heuristic(self, n):
        return n.cost

```

4. Fuzzy Scheduling Constraint Satisfaction Problems

The following code sets up a CSP problem from a given specification.

Hard (unary) domain constraints are applied to reduce the domains of the variables before the constraint solver runs.

```
In [5]: # domain specific CSP builder for week schedule
class CSP_builder():
    # list of text Lines without comments and empty lines
    _, default_domain = Day_Time().string_to_number_set('mon 9am-fri 4pm') # show

    # hard unary constraints: domain is a list of values, params is a single val
    # starts-before, ends-before (for starts-before duration should be 0)
    # vals in domain are actual task start/end date/time, so must be val <= what
    def apply_before(self, param_type, params, duration, domain):
        domain_orig = domain.copy()
        param_val = params.pop()
        for val in domain_orig: # val is week_hour_number
            val1 = val + duration
            h, d = Day_Time().hour_day_split(val1)
            if param_type == 'hour_number' and h > param_val:
                if val in domain: domain.remove(val)
            if param_type == 'day_number' and d > param_val:
                if val in domain: domain.remove(val)
            if param_type == 'week_hour_number' and val1 > param_val:
                if val in domain: domain.remove(val)
        return domain

    def apply_after(self, param_type, params, duration, domain):
        domain_orig = domain.copy()
        param_val = params.pop()
        for val in domain_orig: # val is week_hour_number
            val1 = val + duration
            h, d = Day_Time().hour_day_split(val1)
            if param_type == 'hour_number' and h < param_val:
                if val in domain: domain.remove(val)
            if param_type == 'day_number' and d < param_val:
                if val in domain: domain.remove(val)
            if param_type == 'week_hour_number' and val1 < param_val:
                if val in domain: domain.remove(val)
        return domain

    # day time range only
    # includes starts-in, ends-in
    # duration is 0 for starts-in, task duration for ends-in
    def apply_in(self, params, duration, domain):
        domain_orig = domain.copy()
        for val in domain_orig: # val is week_hour_number
            # task must be within range
            if val in domain and val+duration not in params:
                domain.remove(val)
        return domain

    # task must start at day/time
    def apply_at(self, param_type, param, domain):
        domain_orig = domain.copy()
        for val in domain_orig:
```

```

    h, d = Day_Time().hour_day_split(val)
    if param_type == 'hour_number' and param != h:
        if val in domain: domain.remove(val)
    if param_type == 'day_number' and param != d:
        if val in domain: domain.remove(val)
    if param_type == 'week_hour_number' and param != val:
        if val in domain: domain.remove(val)
    return domain

# soft deadline constraints: return cost to break constraint
# ends-by implementation: domain_dt is the day, hour from the domain
# constr_dt is the soft const spec, dur is the duration of task
# soft_cost is the unit cost of completion delay
# so if the tasks starts on domain_dt, it ends on domain_dt+dur
"""
<t> ends-by <day> <time>, both must be specified
delay = day_hour(T2) - day_hour(T1) + 24*(D2 - D1),
where day_hour(9am) = 0, day_hour(5pm) = 7
"""

def ends_by(self, domain_dt, constr_dt_str, dur, soft_cost):
    param_type, params = Day_Time().string_to_number_set(constr_dt_str)
    param_val = params.pop()
    dom_h, dom_d = Day_Time().hour_day_split(domain_dt+dur)
    if param_type == 'week_hour_number':
        con_h, con_d = Day_Time().hour_day_split(param_val)
        return 0 if domain_dt + dur <= param_val else soft_cost*(dom_h - con_d)
    else:
        return None # not good, must be day and time

def no_cost(self, day, hour):
    return 0

# hard binary constraint, the rest are implemented as gt, lt, eq
def same_day(self, week_hour1, week_hour2):
    h1, d1 = Day_Time().hour_day_split(week_hour1)
    h2, d2 = Day_Time().hour_day_split(week_hour2)
    return d1 == d2

# domain is a list of values
def apply_hard_constraint(self, domain, duration, spec):
    tokens = func_key = spec.split(' ')
    if len(tokens) > 1:
        func_key = spec.split(' ')[0].strip()
    param_type, params = Day_Time().string_to_number_set(spec[len(func_key):])
    if func_key == 'starts-before':
        # duration is 0 for starts before, since we do not modify the time
        return self.apply_before(param_type, params, 0, domain)
    if func_key == 'ends-before':
        return self.apply_before(param_type, params, duration, domain)
    if func_key == 'starts-after':
        return self.apply_after(param_type, params, 0, domain)
    if func_key == 'ends-after':
        return self.apply_after(param_type, params, duration, domain)
    if func_key == 'starts-in':
        return self.apply_in(params, 0, domain)
    if func_key == 'ends-in':
        return self.apply_in(params, duration, domain)
    else:
        # here we have task day or time, it has no func key so we need to parse
        param_type, params = Day_Time().string_to_week_hour_number(spec)

```

```

    return self.apply_at(param_type, params, domain)

def get_cost_function(self, spec):
    func_dict = {'ends-by':self.ends_by, 'no-cost':self.no_cost}
    return [func_dict[spec]]

# spec is the text of a constraint, e.g. 't1 before t2'
# durations are durations of all tasks
def get_binary_constraint(self, spec, durations):
    tokens = spec.strip().split(' ')
    if len(tokens) != 3: return None # error in spec
    # task1 relation task2
    fun_dict = {'before':le, 'after':ge, 'starts-at':eq, 'same-day':self.same_day}
    return Task_Constraint((tokens[0].strip(), tokens[2].strip()), spec, fun_dict)

def get_CSP_with_Cost(self, input_lines):
    # Note: It would be more elegant to make task a class but AIpython is not
    # CSP_with_Cost inherits from CSP, which takes domains and constraints first
    domains = dict()
    constraints = []
    cost_functions = dict()
    durations = dict() # durations of tasks
    soft_day_time = dict() # day time specs of soft constraints
    soft_costs = dict() # costs of soft constraints

    for input_line in input_lines:
        func_spec = None
        input_line_tokens = input_line.strip().split(',')
        if len(input_line_tokens) != 2:
            return None # must have number of tokens = 2
        line_token1 = input_line_tokens[0].strip()
        line_token2 = input_line_tokens[1].strip()
        if line_token1 == 'task':
            tokens = line_token2.split(' ')
            if len(tokens) != 2:
                return None # must have number of tokens = 3
            key = tokens[0].strip()
            # check the duration and save it
            duration = int(tokens[1].strip())
            if duration > Day_Time().num_hours_in_day:
                return None
            durations[key] = duration
            # set zero cost function for this task as default, may add real
            cost_functions[key] = self.get_cost_function('no-cost')
            soft_costs[key] = '0'
            soft_day_time[key] = 'fri 5pm'
            # restrict domain to times that are within allowed range
            # that is start 9-5, start+duration in 9-5
            domains[key] = {x for x in self.default_domain \
                           if Day_Time().day_number(x+duration) \
                           == Day_Time().day_number(x)}
        elif line_token1 == 'domain':
            tokens = line_token2.split(' ')
            if len(tokens) < 2:
                return None # must have number of tokens >= 2
            key = tokens[0].strip()
            # if soft constraint, it is handled differently from hard constraint
            if tokens[1].strip() == 'ends-by':
                # need to retain day time and cost from the line
                # must have task, 'end-by', day, time, cost

```

```

        # or task, 'end-by', day, cost
        # or task, 'end-by', time, cost
        if len(tokens) != 5:
            return None
        # get the rest of the line after 'ends-by'
        soft_costs[key] = int(tokens[len(tokens)-1].strip()) # Last
        # pass the day/time string to avoid passing param_type
        day_time_str = tokens[2] + ' ' + tokens[3]
        soft_day_time[key] = day_time_str
        cost_functions[key] = self.get_cost_function(tokens[1].strip)
    else:
        # the rest of domain spec, after key, are hard unary domain
        # func spec has day/time, we also need duration
        dur = durations[key]
        func_spec = line_token2[len(key):].strip()
        domains[key] = self.apply_hard_constraint(domains[key], dur,
    elif line_token1 == 'constraint': # all binary constraints
        constraints.append(self.get_binary_constraint(line_token2, durat
else:
    return None

return CSP_with_Cost(domains, durations, constraints, cost_functions, so

def create_CSP_from_spec(spec: str):
    input_lines = list()
    spec = spec.split('\n')
    # strip comments
    for input_line in spec:
        input_line = input_line.split('#')
        if len(input_line[0]) > 0:
            input_lines.append(input_line[0])
            print(input_line[0])
    # construct initial CSP problem
    csp = CSP_builder()
    csp_problem = csp.get_CSP_with_Cost(input_lines)
    return csp_problem

```

5. Greedy Search Constraint Solver using Domain Splitting and Arc Consistency

Create a GreedySearcher to search over the CSP.

The *cost* function for CSP nodes is used as the heuristic, but is actually a direct estimate of the total path cost function *f* used in A* Search.

```
In [6]: from searchGeneric import AStarSearcher

class GreedySearcher(AStarSearcher):
    """ returns a searcher for a problem.
    Paths can be found by repeatedly calling search().
    """
    def add_to_frontier(self, path):
        """ add path to the frontier with the appropriate cost """
        # value = path.cost + self.problem.heuristic(path.end()) -- A* definitio
        value = path.end().cost
        self.frontier.add(path, value)
```

Run the GreedySearcher on the CSP derived from the sample input.

Note: The solution cost will always be 0 (which is wrong for the sample input) until you write the cost function in the cell above.

```
In [7]: # Sample problem specification
```

```
sample_spec = """
# two tasks with two binary constraints and soft deadlines
task, t1 3
task, t2 4
# two binary constraints
constraint, t1 before t2
constraint, t1 same-day t2
# domain constraint
domain, t2 mon
# soft deadlines
domain, t1 ends-by mon 3pm 10
domain, t2 ends-by mon 3pm 10
"""
```

```
In [8]: # display details (0 turns off)
```

```
Con_solver.max_display_level = 0
Search_with_AC_from_Cost_CSP.max_display_level = 2
GreedySearcher.max_display_level = 0

def test_csp_solver(searcher):
    final_path = searcher.search()
    if final_path == None:
        print('No solution')
    else:
        domains = final_path.end().domains
        result_str = ''
        for name, domain in domains.items():
            for n in domain:
                result_str += '\n'+str(name)+': '+Day_Time().week_hour_number_to
print(result_str[1:]+'\ncost: '+str(final_path.end().cost))

csp_problem = create_CSP_from_spec(sample_spec)
solver = GreedySearcher(Search_with_AC_from_Cost_CSP(csp_problem))
test_csp_solver(solver)
```

```
task, t1 3
task, t2 4
constraint, t1 before t2
constraint, t1 same-day t2
domain, t2 mon
domain, t1 ends-by mon 3pm 10
domain, t2 ends-by mon 3pm 10
t1: mon 9am
t2: mon 12pm
cost: 10
```

6. Depth-First Search Constraint Solver

The Depth-First Constraint Solver in AI^{Python} by default uses a random ordering of the variables in the CSP.

We need to modify this code to make it compatible with the arc consistency solver.

Run the solver by calling `dfs_solve1` (first solution) or `dfs_solve_all` (all solutions).

```
In [9]: num_expanded = 0
display = False

def dfs_solver(constraints, domains, context, var_order):
    """ generator for all solutions to csp
        context is an assignment of values to some of the variables
        var_order is a list of the variables in csp that are not in context
    """
    global num_expanded, display
    to_eval = {c for c in constraints if c.can_evaluate(context)}
    if all(c.holds(context) for c in to_eval):
        if var_order == []:
            print("Nodes expanded to reach solution:", num_expanded)
            yield context
        else:
            rem_cons = [c for c in constraints if c not in to_eval]
            var = var_order[0]
            for val in domains[var]:
                if display:
                    print("Setting", var, "to", val)
                num_expanded += 1
                yield from dfs_solver(rem_cons, domains, context|{var:val}, var_order)

def dfs_solve_all(csp, var_order=None):
    """ depth-first CSP solver to return a list of all solutions to csp """
    global num_expanded
    num_expanded = 0
    if var_order == None:      # use an arbitrary variable order
        var_order = list(csp.domains)
    return list(dfs_solver(csp.constraints, csp.domains, {}, var_order))

def dfs_solve1(csp, var_order=None):
    """ depth-first CSP solver """
    global num_expanded
    num_expanded = 0
    if var_order == None:      # use an arbitrary variable order
        var_order = list(csp.domains)
    for sol in dfs_solver(csp.constraints, csp.domains, {}, var_order):
        return sol # return first one
```

Run the Depth-First Solver on the sample problem.

Note: Again there are no costs calculated.

```
In [10]: def test_dfs_solver(csp_problem):
    solution = dfs_solve1(csp_problem)
    if solution == None:
        print('No solution')
    else:
        result_str = ''
        for name in solution.keys():
            result_str += '\n'+str(name)+': '+Day_Time().week_hour_number_to_day
    print(result_str[1:])

# call the Depth-First Search solver
```

```

csp_problem = create_CSP_from_spec(sample_spec)
test_dfs_solver(csp_problem) # set display to True to see nodes expanded

task, t1 3
task, t2 4
constraint, t1 before t2
constraint, t1 same-day t2
domain, t2 mon
domain, t1 ends-by mon 3pm 10
domain, t2 ends-by mon 3pm 10
Nodes expanded to reach solution: 5
t1: mon 9am
t2: mon 12pm

```

7. Depth-First Search Constraint Solver using Forward Checking with MRV Heuristic

The Depth-First Constraint Solver in AlPython by default uses a random ordering of the variables in the CSP.

We redefine the `dfs_solver` methods to implement the MRV (Minimum Remaining Values) heuristic using forward checking.

Because the AlPython code is designed to manipulate domain sets, we also need to redefine `can_evaluate` to handle partial assignments.

```

In [11]: num_expanded = 0
display = False

def can_evaluate(c, assignment):
    """ assignment is a variable:value dictionary
        returns True if the constraint can be evaluated given assignment
    """
    return assignment != {} and all(v in assignment.keys() and type(assignment[v]) == set for v in assignment)

def mrv_dfs_solver(constraints, domains, context, var_order):
    """ generator for all solutions to csp.
        context is an assignment of values to some of the variables.
        var_order is a list of the variables in csp that are not in context.
    """
    global num_expanded, display
    if display:
        print("Context", context)
    to_eval = {c for c in constraints if can_evaluate(c, context)}
    if all(c.holds(context) for c in to_eval):
        if var_order == []:
            print("Nodes expanded to reach solution:", num_expanded)
            yield context
        else:
            rem_cons = [c for c in constraints if c not in to_eval] # constraint
            var = var_order[0]
            rem_vars = var_order[1:]
            for val in domains[var]:
                if display:
                    print("Setting", var, "to", val)
                num_expanded += 1
                rem_context = context | {var:val}
                # apply forward checking on remaining variables
                for c in rem_cons:
                    if not c.holds(rem_context):
                        break
                else:
                    yield from mrv_dfs_solver(constraints, domains, rem_context, var_order[1:])

```

```

        if len(var_order) > 1:
            rem_vars_original = list((v, list(domains[v].copy())) for v
            if display:
                print("Original domains:", rem_vars_original)
            # constraints that can't already be evaluated in rem_cons
            rem_cons_ff = [c for c in constraints if c in rem_cons and not
            for rem_var in rem_vars:
                # constraints that can be evaluated by adding a value of
                any_value = list(domains[rem_var])[0]
                rem_to_eval = {c for c in rem_cons_ff if can_evaluate(c,
                # new domain for rem_var are the values for which all ne
                rem_vals = domains[rem_var].copy()
                for rem_val in domains[rem_var]:
                    # no constraint with rem_var in the existing context
                    for c in rem_to_eval:
                        if not c.holds(rem_context|{rem_var: rem_val}):
                            if rem_val in rem_vals:
                                rem_vals.remove(rem_val)
                domains[rem_var] = rem_vals
                # order remaining variables by MRV
                rem_vars.sort(key=lambda v: len(domains[v]))
            if display:
                print("After forward checking:", list((v, domains[v]) for
            if rem_vars == [] or all(len(domains[rem_var]) > 0 for rem_var in
                yield from mrv_dfs_solver(rem_cons, domains, context|{var:v})
            # restore original domains if changed through forward checking
            if len(var_order) > 1:
                if display:
                    print("Restoring original domain", rem_vars_original)
                for (v, domain) in rem_vars_original:
                    domains[v] = domain
            if display:
                print("Nodes expanded so far:", num_expanded)

def mrv_dfs_solve_all(csp, var_order=None):
    """ depth-first CSP solver to return a list of all solutions to csp """
    global num_expanded
    num_expanded = 0
    if var_order == None:    # order variables by MRV
        var_order = list(csp.domains)
        var_order.sort(key=lambda var: len(csp.domains[var]))
    return list(mrv_dfs_solver(csp.constraints, csp.domains, {}, var_order))

def mrv_dfs_solve1(csp, var_order=None):
    """ depth-first CSP solver """
    global num_expanded
    num_expanded = 0
    if var_order == None:    # order variables by MRV
        var_order = list(csp.domains)
        var_order.sort(key=lambda var: len(csp.domains[var]))
    for sol in mrv_dfs_solver(csp.constraints, csp.domains, {}, var_order):
        return sol  # return first one

```

Run this solver on the sample problem.

Note: Again there are no costs calculated.

In [12]: `def test_mrv_dfs_solver(csp_problem):
 solution = mrv_dfs_solve1(csp_problem)`

```

if solution == None:
    print('No solution')
else:
    result_str = ''
    for name in solution.keys():
        result_str += '\n'+str(name)+': '+Day_Time().week_hour_number_to_day
    print(result_str[1:])

# call the Depth-First MRV Search solver
csp_problem = create_CSP_from_spec(sample_spec)
test_mrv_dfs_solver(csp_problem) # set display to True to see nodes expanded

```

```

task, t1 3
task, t2 4
constraint, t1 before t2
constraint, t1 same-day t2
domain, t2 mon
domain, t1 ends-by mon 3pm 10
domain, t2 ends-by mon 3pm 10
Nodes expanded to reach solution: 5
t2: mon 12pm
t1: mon 9am

```

Assignment

Name: Zhiyuan Hu

zID: Z5632280

Question 1 (4 marks)

Consider the search spaces for the fuzzy scheduling CSP solvers – domain splitting with arc consistency and the DFS solver (without forward checking).

- Describe the search spaces in terms of start state, successor functions and goal state(s) (1 mark)
- What is the branching factor and maximum depth to find any solution for the two algorithms (ignoring costs)? (1 mark)
- What is the worst case time and space complexity of the two search algorithms? (1 mark)
- Give one example of a fuzzy scheduling problem that is *easier* for the domain splitting with arc consistency solver than it is for the DFS solver, and explain why (1 mark)

For the second and third part-questions, give the answer in a general form in terms of fuzzy scheduling CSP size parameters.

Answers for Question 1

In a fuzzy scheduling CSP, let n be the number of tasks (variables), d_i the size of the domain for variable i , $d = \max d_i$, and m the number of binary constraints.

For the domain-splitting with arc consistency (AC) method, the start state is a CSP where all unary constraints have been applied and AC has been run once. A successor is created by choosing a variable that is not yet fixed, splitting its domain into two parts, and applying AC again to each part. If the domains are still non-empty, new nodes are generated. The goal state is reached when all variables have domains of size one.

For the depth-first search (DFS) method without forward checking, the start state is an empty assignment. A successor is created by assigning a value to the next variable and continuing the search. The goal is a complete assignment that satisfies all hard constraints.

The maximum branching factor for domain-splitting + AC is 2, because each split produces at most two successors. The maximum depth is about $\sum_{i=1}^n \lceil \log_2 d_i \rceil$, which is at most $n \lceil \log_2 d \rceil$. For DFS, the maximum branching factor is d , and the maximum depth is n .

In terms of time complexity, the worst-case number of nodes for domain-splitting + AC is at most $\prod_{i=1}^n d_i \leq d^n$, and the total time is $O(d^n \cdot md^2)$ because AC takes about $O(md^2)$ per node. The space cost is around $O(nd + m) + O(n \log d)$. For DFS, the worst-case time is $O(d^n)$, and the space is $O(n)$.

A type of problem that is easier for domain-splitting + AC is one with strong structure and strong constraint propagation. For example, if tasks form a chain of ordering constraints like t_1 before t_2 , t_2 before t_3 , and so on, splitting the domain of an early variable allows AC to quickly propagate and remove many impossible values. Many branches are pruned before they are even explored. Pure DFS without forward checking will try many wrong assignments and only detect conflicts deep in the tree, causing a lot of backtracking. This shows why domain-splitting + AC can be much more efficient in such cases.

Question 2 (5 marks)

Define the *cost* function for a fuzzy scheduling CSP (i.e. a node in the search space for domain splitting and arc consistency) as the total cost of the soft deadline constraints violated for all of the variables, assuming that each variable is assigned one of the best possible values from its domain, where a “best” value for a variable v is one that has the lowest cost to violate the soft deadline constraint (if any) for that variable v .

- Implement the cost function in the indicated cell and place a copy of the code below (3 marks)
- What is its computational complexity (give a general form in terms of fuzzy scheduling CSP size parameters)? (1 mark)
- Show that the cost function f never decreases along a path, and explain why this means the search algorithm is optimal (1 mark)

In []: `def calculate_cost(self, context=None, use_lower_bound=True):# context: partial`

```

import math
if context is None:
    context = {}

total_cost = 0

# Iterate over all tasks and their current domains
for task, domain in self.domains.items():
    # If any task's domain has been wiped out by arc consistency (AC),
    # this node is impossible, so return infinity cost.
    if not domain:
        return math.inf

    # Retrieve the cost function associated with this task.
    funcs = self.cost_functions.get(task)
    if not funcs:
        # Safety fallback: if no cost function exists for this task,
        # treat its cost as 0 to avoid errors.
        funcs = [lambda *args, **kwargs: 0]

    duration = self.durations[task]
    deadline = self.soft_day_time[task]
    soft_cost = int(self.soft_costs[task])

    if task in context:
        # Assigned task: compute the real cost using the cost function
        start = context[task]
        total_cost += sum(
            f(start, deadline, duration, soft_cost) for f in funcs
        )
    else:
        # Unassigned task
        if use_lower_bound:
            # Use optimistic lower bound:
            # take the minimum possible soft cost over all current domain values
            best = min(
                sum(f(st, deadline, duration, soft_cost) for f in funcs)
                for st in domain
            )
            total_cost += best
        # if lower bound is not requested, skip unassigned tasks

return total_cost

```

Answers for Question 2

2 The computational complexity of the cost function is

$$O(T \times D \times S)$$

, T is the number of tasks (variables) in the fuzzy scheduling CSP, , D is the average domain size of each task (the number of possible time assignments), , S is the number of soft deadline constraints associated with each task.

The cost function iterates over all tasks, checks each possible value in their domain, and computes the cost for each relevant soft constraint. As a result, the total computation grows linearly with all three parameters.

3 Monotonicity of the Cost Function and Optimality

In a fuzzy scheduling CSP, the cost function (f) measures the total penalty from soft constraint violations under the current partial assignment.

As we go deeper along a search path and assign more tasks, the set of possible completions becomes smaller. Because additional assignments can only add penalties or leave them unchanged, the total cost cannot decrease. Formally, for any node (n_i) and its successor (n_{i+1}):

$$[f(n_{i+1}) > f(n_i)]$$

This means that (f) is **monotonic non-decreasing** along the path.

This property guarantees that if we expand nodes in order of increasing cost (as in a uniform-cost or cost-based search), the first complete solution found must have the minimum possible cost. No cheaper solution can exist deeper in the tree, because all deeper nodes must have cost at least as large as their ancestors.

Therefore, the monotonicity of (f) directly ensures that the search algorithm is **optimal**.

Question 3 (4 marks)

Conduct an empirical evaluation of the domain splitting CSP solver using the cost function defined as above compared to using no cost function (i.e. the zero cost function, as originally defined in the above cell). Use the *average number of nodes expanded* as a metric to compare the two algorithms.

- Write a function `generate_problem(n)` that takes an integer `n` and generates a problem specification with `n` tasks and a random set of hard constraints and soft deadline constraints in the correct format for the constraint solvers (2 marks)

Run the CSP solver (with and without the cost function) over a number of problems of size `n` for a range of values of `n`.

- Plot the performance of the two constraint solving algorithms on the above metric against `n` (1 mark)
- Quantify the performance gain (if any) achieved by the use of this cost function (1 mark)

In [13]:

```
# Random DSL Generator
# After inputting n, a dsl with a scale of n will be returned,
# seed is designed to maintain the reversibility of random results

import random

def generate_problem(n: int, seed=None) -> str:

    if seed is not None:
```

```

random.seed(seed)

days = ["mon", "tue", "wed", "thu", "fri"]
times = ["9am", "10am", "11am", "12pm", "1pm", "2pm", "3pm", "4pm"]
rates = [5, 10, 20]

lines = []
lines.append("# auto-generated spec")

# Tasks and durations
for i in range(1, n + 1):
    dur = random.randint(1, 4)
    lines.append(f"task, t{i} {dur}")

# Random hard constraints between adjacent tasks
for i in range(1, n):
    if random.random() < 0.7:
        lines.append(f"constraint, t{i} before t{i+1}")
    if random.random() < 0.3:
        lines.append(f"constraint, t{i} same-day t{i+1}")

# Soft deadlines (one per task)
for i in range(1, n + 1):
    d = random.choice(days)
    t = random.choice(times)
    r = random.choice(rates)
    lines.append(f"domain, t{i} ends-by {d} {t} {r}")

spec = "\n".join(lines)
return spec

```

Answers for Question 3

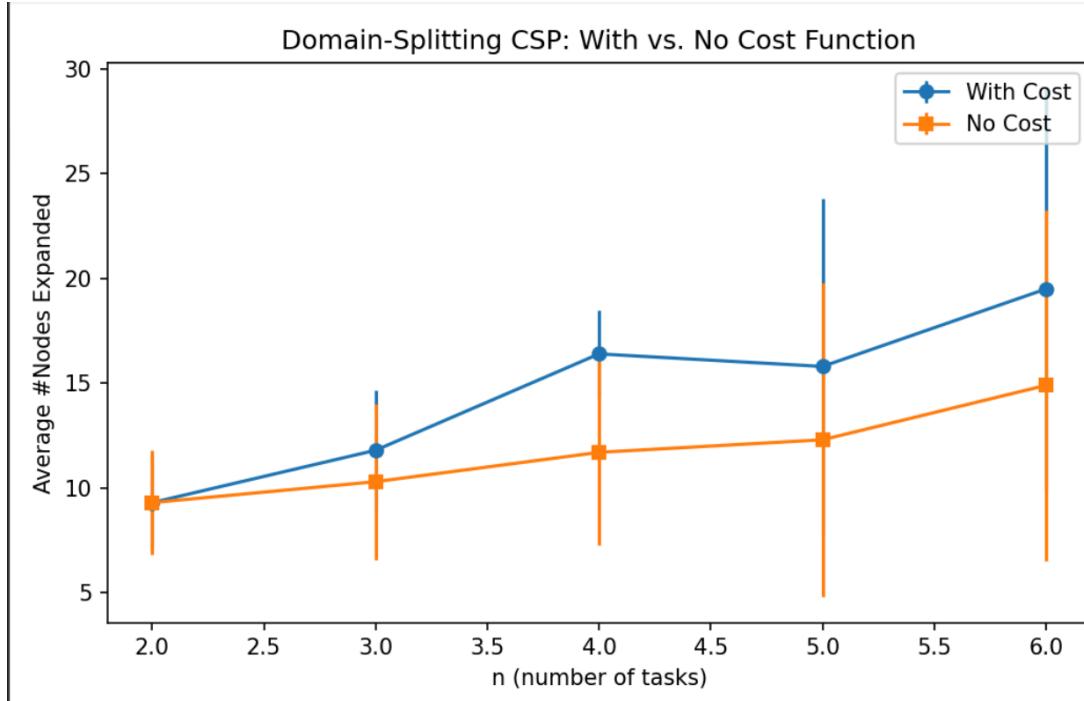
Write the other answers here.

I used GreedySearcher to run the scenarios with cost and without cost respectively, and compared the number of expanded nodes in the two cases. To complete this experiment, I also made some special modifications: I manually added the calculation method of the expansion node. The following is my code and the running result (which includes the average number of expansion nodes for running k times under different n conditions, and the performance gain)

Average Nodes Expanded (TRIALS = 10)

n	with_cost	no_cost	gain(%) = (no - with)/no * 100
2	9.30	9.30	0.0%
3	11.80	10.30	-14.6%
4	16.40	11.70	-40.2%
5	15.80	12.30	-28.5%
6	19.50	14.90	-30.9%

The results show that using a cost function generally leads to **more nodes being expanded** compared to the solver without a cost function. This indicates that the cost-aware solver explores a larger portion of the search space before finding a solution, which reduces search efficiency. For example, when ($n = 4$), the cost-based solver expands about 40% more nodes on average. These results suggest that while a cost function might help guide the solver toward lower-cost solutions, it does not necessarily make the search process more efficient and can even significantly increase the computational effort required.



```
In [ ]: #this is the main code where the result of last cell from
import matplotlib.pyplot as plt

import math
import statistics as stats
import matplotlib.pyplot as plt

def run_search_and_count(search_problem):
    """
    Run Greedy Best-First Search on `search_problem` and count expanded nodes.

    Mechanism:
    - We maintain a frontier (priority queue) of paths; the priority is
      determined by GreedySearcher.add_to_frontier (above).
    - Each iteration pops the best-priority path. Popping is considered
      "expanding" that path's end node; we increment `expanded_nodes`.
    - If the popped path already reaches a goal, we return it.
    - Otherwise we generate all outgoing arcs from the current end node
      via `search_problem.neighbors(node)` and push the successor paths
      back to the frontier.
    - We try to be compatible with multiple Path interfaces (`add_arc`,
      `+` operator, or direct constructor).

    Returns:
        (path_or_None, expanded_nodes)
    """
    frontier = GreedySearcher()
    frontier.add_to_frontier([(None, None, 0)])
    expanded_nodes = 0
    while len(frontier) > 0:
        path = frontier.pop()
        if search_problem.is_goal(path[-1]):
            return (path, expanded_nodes)
        for neighbor in search_problem.neighbors(path[-1]):
            new_path = path + [neighbor]
            if not frontier.contains(new_path):
                frontier.add_to_frontier(new_path)
    return (None, expanded_nodes)
```

```

    path_or_None: the first goal-reaching path found (or None if none).
    expanded_nodes: the number of popped/expanded nodes during the run.
"""

from searchProblem import Path as SearchPath

searcher = GreedySearcher(search_problem)
expanded_nodes = 0

# Some implementations auto-seed the frontier with the start path.
# If not, push the start path here.
try:
    empty = searcher.frontier.empty()
except AttributeError:
    empty = False
if empty:
    searcher.add_to_frontier(SearchPath(search_problem.start_node()))

while not searcher.frontier.empty():
    path = searcher.frontier.pop()
    expanded_nodes += 1 # count an expansion whenever we pop

    # Goal test on the end node of the popped path
    if search_problem.is_goal(path.end()):
        return path, expanded_nodes

    # Push all successor paths induced by outgoing arcs
    for arc in search_problem.neighbors(path.end()):
        # Support several Path APIs
        try:
            new_path = path.add_arc(arc)           # common API: add_arc()
        except AttributeError:
            try:
                new_path = path + arc           # overloaded plus
            except Exception:
                new_path = SearchPath(path, arc) # fallback: explicit ctor
        searcher.add_to_frontier(new_path)

# No solution found
return None, expanded_nodes

def main():

    # settings: a range of n and number of trials per n
    Ns = [2, 3, 4, 5, 6]
    TRIALS = 10

    # aggregates
    avg_with_cost, std_with_cost = [], []
    avg_no_cost, std_no_cost = [], []

    for n in Ns:
        expanded_w_list = []
        expanded_0_list = []

        # multiple random trials per n
        for trial in range(TRIALS):
            spec = generate_problem(n=n, seed=1000*n + trial)

            # --- WITH cost ---

```

```

csp_w = create_CSP_from_spec(spec)                                # parse DSL → CS
prob_w = Search_with_AC_from_Cost_CSP(csp_w)                      # build search p
_, exp_w = run_search_and_count(prob_w)                            # run greedy & c
expanded_w_list.append(exp_w)

# --- NO cost: replace cost_functions with zero-cost Lambdas ---
csp_0 = create_CSP_from_spec(spec)
for k in csp_0.cost_functions.keys():
    csp_0.cost_functions[k] = [lambda *args, **kwargs: 0]
prob_0 = Search_with_AC_from_Cost_CSP(csp_0)
_, exp_0 = run_search_and_count(prob_0)
expanded_0_list.append(exp_0)

# aggregate
avg_with_cost.append(stats.mean(expanded_w_list))
std_with_cost.append(stats.pstdev(expanded_w_list))
avg_no_cost.append(stats.mean(expanded_0_list))
std_no_cost.append(stats.pstdev(expanded_0_list))

# ===== print summary =====
print("\n==== Average Nodes Expanded (TRIALS = {}) ====".format(TRIALS))
print(" n | with_cost | no_cost | gain(%) = (no - with)/no * 100")
for i, n in enumerate(Ns):
    with_avg = avg_with_cost[i]
    no_avg = avg_no_cost[i]
    gain = 0.0 if no_avg == 0 else (no_avg - with_avg) / no_avg * 100.0
    print("{:2d} | {:9.2f} | {:7.2f} | {:6.1f}%".format(n, with_avg, no_avg,
                                                       gain))

# ===== plot (two curves with error bars) =====
plt.figure(figsize=(7, 4.5))
plt.errorbar(Ns, avg_with_cost, yerr=std_with_cost, marker='o', label='With')
plt.errorbar(Ns, avg_no_cost, yerr=std_no_cost, marker='s', label='No Co')
plt.xlabel("n (number of tasks)")
plt.ylabel("Average #Nodes Expanded")
plt.title("Greedy Search: With vs. No Cost Function")
plt.legend()
plt.tight_layout()
plt.show()

# ===== entry point =====
if __name__ == "__main__":
    main()

```

```
task, t1 4
task, t2 1
domain, t1 ends-by wed 11am 20
domain, t2 ends-by tue 3pm 5
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t1 into {0, 1, 2, 3, 8} and {9, 10, 11, 16, 17}
Splitting t1 into {9, 10} and {16, 17, 11}
Splitting t1 into {16} and {17, 11}
Splitting t1 into {17} and {11}
Splitting t2 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t2 into {0, 1, 2, 3, 4, 5, 6, 8} and {9, 10, 11, 12, 13, 14, 16, 17, 18}
Splitting t2 into {9, 10, 11, 12} and {13, 14, 16, 17, 18}
Splitting t2 into {13, 14} and {16, 17, 18}
Splitting t2 into {13} and {14}
task, t1 4
task, t2 1
domain, t1 ends-by wed 11am 20
domain, t2 ends-by tue 3pm 5
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t1 into {32, 33, 34, 35, 18} and {19, 24, 25, 26, 27}
Splitting t1 into {24, 19} and {25, 26, 27}
Splitting t1 into {25} and {26, 27}
Splitting t1 into {26} and {27}
Splitting t2 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t2 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t2 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t2 into {26, 27} and {28, 29, 30}
Splitting t2 into {28} and {29, 30}
Splitting t2 into {29} and {30}
task, t1 1
task, t2 3
constraint, t1 same-day t2
domain, t1 ends-by fri 9am 20
domain, t2 ends-by tue 9am 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8} and {9, 10, 11, 12, 13, 14, 16, 17, 18}
Splitting t1 into {0, 1, 2, 3} and {8, 4, 5, 6}
Splitting t1 into {8, 4} and {5, 6}
Splitting t1 into {5} and {6}
Splitting t2 into {0, 1} and {2, 3, 4}
Splitting t2 into {2} and {3, 4}
Splitting t2 into {3} and {4}
task, t1 1
task, t2 3
constraint, t1 same-day t2
domain, t1 ends-by fri 9am 20
domain, t2 ends-by tue 9am 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
```

```

Splitting t1 into {26, 27} and {28, 29, 30}
Splitting t1 into {28} and {29, 30}
Splitting t1 into {29} and {30}
Splitting t2 into {24, 25} and {26, 27, 28}
Splitting t2 into {26} and {27, 28}
Splitting t2 into {27} and {28}
task, t1 1
task, t2 3
domain, t1 ends-by wed 1pm 20
domain, t2 ends-by mon 10am 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and
{32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 2
7, 28, 29, 30}
Splitting t1 into {32, 33, 34, 35} and {36, 37, 38, 19, 20}
Splitting t1 into {36, 37} and {19, 20, 38}
Splitting t1 into {19} and {20, 38}
Splitting t2 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35,
36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t2 into {0, 1, 2, 3, 4, 8} and {9, 10, 11, 12, 16, 17}
Splitting t2 into {0, 1, 2} and {8, 3, 4}
Splitting t2 into {0} and {1, 2}
task, t1 1
task, t2 3
domain, t1 ends-by wed 1pm 20
domain, t2 ends-by mon 10am 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and
{32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 2
7, 28, 29, 30}
Splitting t1 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t1 into {26, 27} and {28, 29, 30}
Splitting t1 into {28} and {29, 30}
Splitting t1 into {29} and {30}
Splitting t2 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35,
36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t2 into {32, 33, 34, 35, 36, 18} and {19, 20, 24, 25, 26, 27, 28}
Splitting t2 into {24, 19, 20} and {25, 26, 27, 28}
Splitting t2 into {25, 26} and {27, 28}
Splitting t2 into {27} and {28}
task, t1 1
task, t2 2
constraint, t1 before t2
domain, t1 ends-by mon 11am 10
domain, t2 ends-by wed 10am 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17} and {3
2, 33, 34, 35, 36, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8} and {9, 10, 11, 12, 13, 14, 16, 17}
Splitting t1 into {0, 1, 2, 3} and {8, 4, 5, 6}
Splitting t1 into {0, 1} and {2, 3}
Splitting t1 into {0} and {1}
Splitting t2 into {2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18, 19} and {32, 33,
34, 35, 36, 37, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t2 into {2, 3, 4, 5, 8, 9, 10} and {11, 12, 13, 16, 17, 18, 19}
Splitting t2 into {11, 12, 13} and {16, 17, 18, 19}
Splitting t2 into {11} and {12, 13}
Splitting t2 into {12} and {13}
task, t1 1
task, t2 2
constraint, t1 before t2

```

```
domain, t1 ends-by mon 11am 10
domain, t2 ends-by wed 10am 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17} and {3
2, 33, 34, 35, 36, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {32, 33, 34, 35, 36, 18, 19, 20} and {21, 22, 24, 25, 26, 27, 2
8, 29, 30}
Splitting t1 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t1 into {26, 27} and {28, 29, 30}
Splitting t1 into {28} and {29, 30}
Splitting t1 into {29} and {30}
Splitting t2 into {32, 33, 34} and {35, 36, 37}
Splitting t2 into {35} and {36, 37}
Splitting t2 into {36} and {37}
task, t1 1
task, t2 2
constraint, t1 same-day t2
domain, t1 ends-by fri 11am 10
domain, t2 ends-by tue 11am 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and
{32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8} and {9, 10, 11, 12, 13, 14, 16, 17, 1
8}
Splitting t1 into {9, 10, 11, 12} and {13, 14, 16, 17, 18}
Splitting t1 into {13, 14} and {16, 17, 18}
Splitting t1 into {13} and {14}
Splitting t2 into {8, 9, 10} and {11, 12, 13}
Splitting t2 into {8} and {9, 10}
task, t1 1
task, t2 2
constraint, t1 same-day t2
domain, t1 ends-by fri 11am 10
domain, t2 ends-by tue 11am 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and
{32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 2
7, 28, 29, 30}
Splitting t1 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t1 into {26, 27} and {28, 29, 30}
Splitting t1 into {28} and {29, 30}
Splitting t1 into {29} and {30}
Splitting t2 into {24, 25, 26} and {27, 28, 29}
Splitting t2 into {27} and {28, 29}
Splitting t2 into {28} and {29}
task, t1 4
task, t2 1
constraint, t1 before t2
constraint, t1 same-day t2
domain, t1 ends-by thu 10am 5
domain, t2 ends-by thu 2pm 10
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11, 16} and {32, 33, 34, 17, 18, 19, 24,
25, 26, 27}
Splitting t1 into {32, 33, 34, 17, 18} and {19, 24, 25, 26, 27}
Splitting t1 into {32, 33} and {17, 34, 18}
Splitting t1 into {17} and {34, 18}
Splitting t1 into {34} and {18}
task, t1 4
task, t2 1
constraint, t1 before t2
constraint, t1 same-day t2
domain, t1 ends-by thu 10am 5
```

```

domain, t2 ends-by thu 2pm 10
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11, 16} and {32, 33, 34, 17, 18, 19, 24, 25, 26, 27}
Splitting t1 into {32, 33, 34, 17, 18} and {19, 24, 25, 26, 27}
Splitting t1 into {24} and {25, 26}
Splitting t1 into {25} and {26}
task, t1 4
task, t2 1
constraint, t1 before t2
constraint, t1 same-day t2
domain, t1 ends-by fri 10am 20
domain, t2 ends-by tue 12pm 10
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11, 16} and {32, 33, 34, 17, 18, 19, 24, 25, 26, 27}
Splitting t1 into {0, 1, 2, 3} and {8, 9, 10, 11, 16}
Splitting t1 into {0} and {1, 2}
Splitting t1 into {1} and {2}
task, t1 4
task, t2 1
constraint, t1 before t2
constraint, t1 same-day t2
domain, t1 ends-by fri 10am 20
domain, t2 ends-by tue 12pm 10
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11, 16} and {32, 33, 34, 17, 18, 19, 24, 25, 26, 27}
Splitting t1 into {32, 33, 34, 17, 18} and {19, 24, 25, 26, 27}
Splitting t1 into {24} and {25, 26}
Splitting t1 into {25} and {26}
task, t1 2
task, t2 1
domain, t1 ends-by fri 2pm 5
domain, t2 ends-by fri 11am 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t1 into {26, 27} and {28, 29}
Splitting t1 into {28} and {29}
Splitting t2 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t2 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t2 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t2 into {26, 27} and {28, 29, 30}
Splitting t2 into {28} and {29, 30}
Splitting t2 into {29} and {30}
task, t1 2
task, t2 1
domain, t1 ends-by fri 2pm 5
domain, t2 ends-by fri 11am 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t1 into {26, 27} and {28, 29}
Splitting t1 into {28} and {29}
Splitting t2 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}

```

Splitting t2 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t2 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t2 into {26, 27} and {28, 29, 30}
Splitting t2 into {28} and {29, 30}
Splitting t2 into {29} and {30}
task, t1 2
task, t2 4
constraint, t1 before t2
domain, t1 ends-by thu 3pm 10
domain, t2 ends-by tue 2pm 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {0, 1, 2, 3, 4, 5} and {8, 9, 10, 11, 12, 13, 16}
Splitting t1 into {0, 1, 2} and {3, 4, 5}
Splitting t1 into {3} and {4, 5}
Splitting t1 into {4} and {5}
Splitting t2 into {32, 33, 34, 35, 8, 9, 10, 11} and {16, 17, 18, 19, 24, 25, 26, 27}
Splitting t2 into {32, 33, 34, 35} and {8, 9, 10, 11}
Splitting t2 into {8, 9} and {10, 11}
Splitting t2 into {8} and {9}
task, t1 2
task, t2 4
constraint, t1 before t2
domain, t1 ends-by thu 3pm 10
domain, t2 ends-by tue 2pm 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 17, 18, 19, 20} and {21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {24, 25, 21} and {26, 27, 28, 29}
Splitting t1 into {26, 27} and {28, 29}
Splitting t1 into {28} and {29}
Splitting t2 into {32, 33} and {34, 35}
Splitting t2 into {34} and {35}
task, t1 3
task, t2 2
constraint, t1 before t2
domain, t1 ends-by wed 11am 10
domain, t2 ends-by fri 10am 5
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16} and {32, 33, 34, 17, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t1 into {0, 1, 2, 3, 4} and {8, 9, 10, 11, 12, 16}
Splitting t1 into {8, 9, 10} and {16, 11, 12}
Splitting t1 into {16} and {11, 12}
Splitting t1 into {11} and {12}
Splitting t2 into {32, 33, 34, 35, 36, 37, 16, 17, 18} and {19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t2 into {24, 19, 20, 21} and {25, 26, 27, 28, 29}
Splitting t2 into {25, 26} and {27, 28, 29}
Splitting t2 into {27} and {28, 29}
Splitting t2 into {28} and {29}
task, t1 3
task, t2 2
constraint, t1 before t2
domain, t1 ends-by wed 11am 10
domain, t2 ends-by fri 10am 5
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16} and {32, 33, 34, 17, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t1 into {32, 33, 34, 17, 18, 19} and {20, 24, 25, 26, 27, 28}

Splitting t1 into {24, 25, 20} and {26, 27, 28}
Splitting t1 into {26} and {27, 28}
Splitting t1 into {27} and {28}
Splitting t2 into {32, 33, 34} and {35, 36, 37}
Splitting t2 into {35} and {36, 37}
Splitting t2 into {36} and {37}
task, t1 2
task, t2 3
task, t3 1
constraint, t1 before t2
constraint, t2 before t3
domain, t1 ends-by thu 1pm 5
domain, t2 ends-by tue 12pm 20
domain, t3 ends-by tue 2pm 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {0, 1, 2, 3, 4, 5} and {8, 9, 10, 11, 12, 13, 16}
Splitting t1 into {0, 1, 2} and {3, 4, 5}
Splitting t1 into {3} and {4, 5}
Splitting t1 into {4} and {5}
Splitting t2 into {8, 9, 10, 11, 12, 16, 17, 18, 19} and {32, 33, 34, 35, 20, 24, 25, 26, 27, 28}
Splitting t2 into {8, 9, 10, 11} and {12, 16, 17, 18, 19}
Splitting t2 into {8, 9} and {10, 11}
Splitting t2 into {8} and {9}
Splitting t3 into {11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 24} and {32, 33, 34, 35, 36, 37, 38, 25, 26, 27, 28, 29, 30}
Splitting t3 into {11, 12, 13, 14, 16, 17} and {18, 19, 20, 21, 22, 24}
Splitting t3 into {11, 12, 13} and {16, 17, 14}
Splitting t3 into {11} and {12, 13}
Splitting t3 into {12} and {13}
task, t1 2
task, t2 3
task, t3 1
constraint, t1 before t2
constraint, t2 before t3
domain, t1 ends-by thu 1pm 5
domain, t2 ends-by tue 12pm 20
domain, t3 ends-by tue 2pm 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 17, 18, 19, 20} and {21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {24, 25, 21} and {26, 27, 28, 29}
Splitting t1 into {26, 27} and {28, 29}
Splitting t1 into {28} and {29}
Splitting t2 into {32, 33} and {34, 35}
Splitting t2 into {34} and {35}
task, t1 3
task, t2 4
task, t3 2
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
domain, t1 ends-by wed 4pm 10
domain, t2 ends-by thu 2pm 5
domain, t3 ends-by mon 9am 20
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10} and {11, 12, 16, 17, 18, 19, 20, 24}
Splitting t1 into {0, 1, 2} and {8, 3, 4}
Splitting t3 into {8, 9, 10, 11, 12, 13, 16, 17, 18, 19, 20, 21} and {32, 33, 34, 35, 36, 37, 24, 25, 26, 27, 28, 29}

Splitting t3 into {8, 9, 10, 11, 12, 13} and {16, 17, 18, 19, 20, 21}
Splitting t3 into {8, 9, 10} and {11, 12, 13}
Splitting t3 into {8} and {9, 10}
task, t1 3
task, t2 4
task, t3 2
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
domain, t1 ends-by wed 4pm 10
domain, t2 ends-by thu 2pm 5
domain, t3 ends-by mon 9am 20
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10} and {11, 12, 16, 17, 18, 19, 20, 24}
Splitting t1 into {16, 17, 18} and {24, 19, 20}
Splitting t3 into {32, 33, 34} and {35, 36, 37}
Splitting t3 into {35} and {36, 37}
Splitting t3 into {36} and {37}
task, t1 2
task, t2 4
task, t3 1
constraint, t1 before t2
constraint, t2 before t3
domain, t1 ends-by mon 11am 10
domain, t2 ends-by mon 2pm 10
domain, t3 ends-by mon 3pm 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13} and {32, 16, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {0, 1, 2, 3, 4, 5} and {8, 9, 10, 11, 12, 13}
Splitting t1 into {0, 1, 2} and {3, 4, 5}
Splitting t1 into {0} and {1, 2}
Splitting t2 into {32, 33, 2, 3, 34, 8, 9, 10} and {11, 16, 17, 18, 19, 24, 25, 26, 27}
Splitting t2 into {32, 33, 2, 3} and {8, 9, 34, 10}
Splitting t2 into {32, 33} and {2, 3}
Splitting t2 into {2} and {3}
Splitting t3 into {6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21} and {32, 33, 34, 35, 36, 37, 38, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t3 into {6, 8, 9, 10, 11, 12, 13} and {14, 16, 17, 18, 19, 20, 21}
Splitting t3 into {8, 9, 6} and {10, 11, 12, 13}
Splitting t3 into {8} and {9, 6}
Splitting t3 into {9} and {6}
task, t1 2
task, t2 4
task, t3 1
constraint, t1 before t2
constraint, t2 before t3
domain, t1 ends-by mon 11am 10
domain, t2 ends-by mon 2pm 10
domain, t3 ends-by mon 3pm 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13} and {32, 16, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 16, 17, 18, 19, 20} and {21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {24, 25, 21} and {26, 27, 28, 29}
Splitting t1 into {26, 27} and {28, 29}
Splitting t1 into {28} and {29}
Splitting t2 into {32} and {33, 34}
Splitting t2 into {33} and {34}
task, t1 4
task, t2 2
task, t3 3

```
constraint, t1 before t2
constraint, t2 same-day t3
domain, t1 ends-by wed 9am 5
domain, t2 ends-by tue 11am 5
domain, t3 ends-by mon 11am 5
Splitting t1 into {0, 1, 2, 3, 32, 33, 8, 9, 10} and {11, 16, 17, 18, 19, 24, 25, 26, 27}
Splitting t1 into {0, 1, 2, 3} and {32, 33, 8, 9, 10}
Splitting t1 into {0, 1} and {2, 3}
Splitting t1 into {0} and {1}
Splitting t2 into {5, 8, 9, 10, 11, 12, 13, 16, 17, 18, 19, 20} and {32, 33, 34, 35, 36, 37, 21, 24, 25, 26, 27, 28, 29}
Splitting t2 into {5, 8, 9, 10, 11, 12} and {13, 16, 17, 18, 19, 20}
Splitting t2 into {8, 9, 5} and {10, 11, 12}
Splitting t2 into {8} and {9, 5}
Splitting t2 into {9} and {5}
Splitting t3 into {0, 1} and {2, 3, 4}
Splitting t3 into {0} and {1}
task, t1 4
task, t2 2
task, t3 3
constraint, t1 before t2
constraint, t2 same-day t3
domain, t1 ends-by wed 9am 5
domain, t2 ends-by tue 11am 5
domain, t3 ends-by mon 11am 5
Splitting t1 into {0, 1, 2, 3, 32, 33, 8, 9, 10} and {11, 16, 17, 18, 19, 24, 25, 26, 27}
Splitting t1 into {16, 17, 18, 11} and {19, 24, 25, 26, 27}
Splitting t1 into {24, 19} and {25, 26, 27}
Splitting t1 into {25} and {26, 27}
Splitting t1 into {26} and {27}
Splitting t2 into {32, 33, 34} and {35, 36, 37}
Splitting t2 into {35} and {36, 37}
Splitting t2 into {36} and {37}
Splitting t3 into {32, 33} and {34, 35, 36}
Splitting t3 into {34} and {35, 36}
Splitting t3 into {35} and {36}
task, t1 4
task, t2 4
task, t3 2
constraint, t2 same-day t3
domain, t1 ends-by mon 9am 20
domain, t2 ends-by wed 9am 20
domain, t3 ends-by tue 3pm 10
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t1 into {0, 1, 2, 3, 8} and {9, 10, 11, 16, 17}
Splitting t1 into {0, 1} and {8, 2, 3}
Splitting t1 into {0} and {1}
Splitting t2 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t2 into {0, 1, 2, 3, 8} and {9, 10, 11, 16, 17}
Splitting t2 into {9, 10} and {16, 17, 11}
Splitting t2 into {16} and {17, 11}
Splitting t2 into {17} and {11}
Splitting t3 into {8, 9, 10} and {11, 12, 13}
Splitting t3 into {11} and {12, 13}
Splitting t3 into {12} and {13}
task, t1 4
```

```
task, t2 4
task, t3 2
constraint, t2 same-day t3
domain, t1 ends-by mon 9am 20
domain, t2 ends-by wed 9am 20
domain, t3 ends-by tue 3pm 10
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t1 into {32, 33, 34, 35, 18} and {19, 24, 25, 26, 27}
Splitting t1 into {24, 19} and {25, 26, 27}
Splitting t1 into {25} and {26, 27}
Splitting t1 into {26} and {27}
Splitting t2 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t2 into {32, 33, 34, 35, 18} and {19, 24, 25, 26, 27}
Splitting t2 into {24, 19} and {25, 26, 27}
Splitting t2 into {25} and {26, 27}
Splitting t2 into {26} and {27}
Splitting t3 into {24, 25, 26} and {27, 28, 29}
Splitting t3 into {27} and {28, 29}
Splitting t3 into {28} and {29}
task, t1 1
task, t2 4
task, t3 3
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 same-day t3
domain, t1 ends-by fri 2pm 10
domain, t2 ends-by mon 4pm 5
domain, t3 ends-by fri 9am 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16} and {32, 33, 34, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {0, 1, 2, 3, 4, 5, 6} and {8, 9, 10, 11, 12, 13, 14, 16}
Splitting t1 into {0} and {1, 2}
Splitting t1 into {1} and {2}
Splitting t3 into {0, 1} and {2, 3, 4}
Splitting t3 into {2} and {3, 4}
Splitting t3 into {3} and {4}
task, t1 1
task, t2 4
task, t3 3
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 same-day t3
domain, t1 ends-by fri 2pm 10
domain, t2 ends-by mon 4pm 5
domain, t3 ends-by fri 9am 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16} and {32, 33, 34, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {32, 33, 34, 17, 18, 19, 20, 21} and {22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {24} and {25, 26}
Splitting t1 into {25} and {26}
Splitting t3 into {24, 25} and {26, 27, 28}
Splitting t3 into {26} and {27, 28}
Splitting t3 into {27} and {28}
task, t1 1
task, t2 3
task, t3 3
constraint, t1 same-day t2
```

```

domain, t1 ends-by fri 2pm 10
domain, t2 ends-by mon 2pm 10
domain, t3 ends-by fri 2pm 5
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and
{32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8} and {9, 10, 11, 12, 13, 14, 16, 17, 1
8}
Splitting t1 into {0, 1, 2, 3} and {8, 4, 5, 6}
Splitting t1 into {8, 4} and {5, 6}
Splitting t1 into {5} and {6}
Splitting t2 into {0, 1} and {2, 3, 4}
Splitting t2 into {2} and {3, 4}
Splitting t3 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35,
36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t3 into {32, 33, 34, 35, 36, 18} and {19, 20, 24, 25, 26, 27, 28}
Splitting t3 into {24, 19, 20} and {25, 26, 27, 28}
Splitting t3 into {25, 26} and {27, 28}
Splitting t3 into {27} and {28}
task, t1 1
task, t2 3
task, t3 3
constraint, t1 same-day t2
domain, t1 ends-by fri 2pm 10
domain, t2 ends-by mon 2pm 10
domain, t3 ends-by fri 2pm 5
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and
{32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 2
7, 28, 29, 30}
Splitting t1 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t1 into {26, 27} and {28, 29, 30}
Splitting t1 into {28} and {29, 30}
Splitting t1 into {29} and {30}
Splitting t2 into {24, 25} and {26, 27, 28}
Splitting t2 into {26} and {27, 28}
Splitting t2 into {27} and {28}
Splitting t3 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35,
36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t3 into {32, 33, 34, 35, 36, 18} and {19, 20, 24, 25, 26, 27, 28}
Splitting t3 into {24, 19, 20} and {25, 26, 27, 28}
Splitting t3 into {25, 26} and {27, 28}
Splitting t3 into {27} and {28}
task, t1 3
task, t2 4
task, t3 4
constraint, t1 before t2
constraint, t2 before t3
domain, t1 ends-by mon 2pm 10
domain, t2 ends-by mon 12pm 5
domain, t3 ends-by wed 4pm 5
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10} and {11, 12, 16, 17, 18, 19, 20, 24}
Splitting t1 into {0, 1, 2, 3} and {8, 9, 10, 4}
Splitting t1 into {0, 1} and {2, 3}
Splitting t1 into {0} and {1}
Splitting t2 into {3, 8, 9, 10, 11, 16} and {17, 18, 19, 24, 25, 26, 27}
Splitting t2 into {8, 9, 3} and {16, 10, 11}
Splitting t2 into {8} and {9, 3}
Splitting t2 into {9} and {3}
Splitting t3 into {32, 33, 34, 35, 8, 9, 10, 11} and {16, 17, 18, 19, 24, 25, 26,
27}

```

Splitting t3 into {16, 17, 18, 19} and {24, 25, 26, 27}
Splitting t3 into {16, 17} and {18, 19}
Splitting t3 into {18} and {19}
task, t1 3
task, t2 4
task, t3 4
constraint, t1 before t2
constraint, t2 before t3
domain, t1 ends-by mon 2pm 10
domain, t2 ends-by mon 12pm 5
domain, t3 ends-by wed 4pm 5
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10} and {11, 12, 16, 17, 18, 19, 20, 24}
Splitting t1 into {16, 17, 11, 12} and {24, 18, 19, 20}
Splitting t1 into {24, 18} and {19, 20}
Splitting t1 into {19} and {20}
Splitting t2 into {24, 25} and {26, 27}
Splitting t2 into {26} and {27}
Splitting t3 into {32, 33} and {34, 35}
Splitting t3 into {34} and {35}
task, t1 3
task, t2 4
task, t3 1
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
domain, t1 ends-by mon 2pm 5
domain, t2 ends-by fri 11am 20
domain, t3 ends-by thu 3pm 5
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10} and {11, 12, 16, 17, 18, 19, 20, 24}
Splitting t1 into {0, 1, 2} and {8, 3, 4}
Splitting t3 into {8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22} and {32, 33, 34, 35, 36, 37, 38, 24, 25, 26, 27, 28, 29, 30}
Splitting t3 into {32, 33, 34, 35, 36, 37, 38} and {24, 25, 26, 27, 28, 29, 30}
Splitting t3 into {24, 25, 26} and {27, 28, 29, 30}
Splitting t3 into {27, 28} and {29, 30}
Splitting t3 into {29} and {30}
task, t1 3
task, t2 4
task, t3 1
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
domain, t1 ends-by mon 2pm 5
domain, t2 ends-by fri 11am 20
domain, t3 ends-by thu 3pm 5
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10} and {11, 12, 16, 17, 18, 19, 20, 24}
Splitting t1 into {16, 17, 18} and {24, 19, 20}
Splitting t3 into {32, 33, 34} and {35, 36, 37, 38}
Splitting t3 into {35, 36} and {37, 38}
Splitting t3 into {37} and {38}
task, t1 4
task, t2 3
task, t3 1
domain, t1 ends-by mon 1pm 5
domain, t2 ends-by wed 9am 20
domain, t3 ends-by mon 4pm 20
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t1 into {0, 1, 2, 3, 8} and {9, 10, 11, 16, 17}
Splitting t1 into {0, 1} and {8, 2, 3}

Splitting t1 into {0} and {1}
Splitting t2 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35, 36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t2 into {0, 1, 2, 3, 4, 8} and {9, 10, 11, 12, 16, 17}
Splitting t2 into {9, 10, 11} and {16, 17, 12}
Splitting t2 into {16} and {17, 12}
Splitting t2 into {17} and {12}
Splitting t3 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t3 into {0, 1, 2, 3, 4, 5, 6, 8} and {9, 10, 11, 12, 13, 14, 16, 17, 18}
Splitting t3 into {0, 1, 2, 3} and {8, 4, 5, 6}
Splitting t3 into {8, 4} and {5, 6}
Splitting t3 into {5} and {6}
task, t1 4
task, t2 3
task, t3 1
domain, t1 ends-by mon 1pm 5
domain, t2 ends-by wed 9am 20
domain, t3 ends-by mon 4pm 20
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t1 into {32, 33, 34, 35, 18} and {19, 24, 25, 26, 27}
Splitting t1 into {24, 19} and {25, 26, 27}
Splitting t1 into {25} and {26, 27}
Splitting t1 into {26} and {27}
Splitting t2 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35, 36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t2 into {32, 33, 34, 35, 36, 18} and {19, 20, 24, 25, 26, 27, 28}
Splitting t2 into {24, 19, 20} and {25, 26, 27, 28}
Splitting t2 into {25, 26} and {27, 28}
Splitting t2 into {27} and {28}
Splitting t3 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t3 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t3 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t3 into {26, 27} and {28, 29, 30}
Splitting t3 into {28} and {29, 30}
Splitting t3 into {29} and {30}
task, t1 4
task, t2 4
task, t3 4
task, t4 2
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 before t4
domain, t1 ends-by wed 10am 5
domain, t2 ends-by thu 2pm 5
domain, t3 ends-by mon 12pm 10
domain, t4 ends-by wed 4pm 10
Splitting t1 into {0, 1, 2, 3, 8, 9} and {10, 11, 16, 17, 18, 19}
Splitting t1 into {0, 1, 2} and {8, 9, 3}
Splitting t1 into {8} and {9, 3}
Splitting t1 into {9} and {3}
Splitting t2 into {8, 9, 10, 11, 16, 17} and {18, 19, 24, 25, 26, 27}
Splitting t2 into {8, 9, 10} and {16, 17, 11}
Splitting t2 into {16} and {17, 11}
Splitting t2 into {17} and {11}
Splitting t3 into {32, 33, 16, 17, 18} and {19, 24, 25, 26, 27}

Splitting t3 into {32, 33} and {16, 17, 18}
Splitting t3 into {16} and {17, 18}
Splitting t4 into {32, 33, 34, 35, 36, 37, 20} and {21, 24, 25, 26, 27, 28, 29}
Splitting t4 into {24, 25, 21} and {26, 27, 28, 29}
Splitting t4 into {24} and {25, 21}
Splitting t4 into {25} and {21}
task, t1 4
task, t2 4
task, t3 4
task, t4 2
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 before t4
domain, t1 ends-by wed 10am 5
domain, t2 ends-by thu 2pm 5
domain, t3 ends-by mon 12pm 10
domain, t4 ends-by wed 4pm 10
Splitting t1 into {0, 1, 2, 3, 8, 9} and {10, 11, 16, 17, 18, 19}
Splitting t1 into {16, 10, 11} and {17, 18, 19}
Splitting t1 into {17} and {18, 19}
Splitting t1 into {18} and {19}
Splitting t2 into {24, 25} and {26, 27}
Splitting t2 into {26} and {27}
Splitting t3 into {32} and {33}
task, t1 4
task, t2 2
task, t3 4
task, t4 2
constraint, t1 before t2
domain, t1 ends-by fri 12pm 10
domain, t2 ends-by fri 4pm 10
domain, t3 ends-by wed 11am 5
domain, t4 ends-by mon 4pm 5
Splitting t1 into {0, 1, 2, 3, 32, 33, 8, 9, 10} and {11, 16, 17, 18, 19, 24, 25, 26, 27}
Splitting t1 into {16, 17, 18, 11} and {19, 24, 25, 26, 27}
Splitting t1 into {24, 19} and {25, 26, 27}
Splitting t1 into {25} and {26, 27}
Splitting t1 into {26} and {27}
Splitting t2 into {32, 33, 34} and {35, 36, 37}
Splitting t2 into {35} and {36, 37}
Splitting t2 into {36} and {37}
Splitting t3 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t3 into {0, 1, 2, 3, 8} and {9, 10, 11, 16, 17}
Splitting t3 into {9, 10} and {16, 17, 11}
Splitting t3 into {16} and {17, 11}
Splitting t3 into {17} and {11}
Splitting t4 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t4 into {0, 1, 2, 3, 4, 5, 8} and {9, 10, 11, 12, 13, 16, 17, 18}
Splitting t4 into {0, 1, 2} and {8, 3, 4, 5}
Splitting t4 into {8, 3} and {4, 5}
Splitting t4 into {4} and {5}
task, t1 4
task, t2 2
task, t3 4
task, t4 2
constraint, t1 before t2
domain, t1 ends-by fri 12pm 10

```
domain, t2 ends-by fri 4pm 10
domain, t3 ends-by wed 11am 5
domain, t4 ends-by mon 4pm 5
Splitting t1 into {0, 1, 2, 3, 32, 33, 8, 9, 10} and {11, 16, 17, 18, 19, 24, 25, 26, 27}
Splitting t1 into {16, 17, 18, 11} and {19, 24, 25, 26, 27}
Splitting t1 into {24, 19} and {25, 26, 27}
Splitting t1 into {25} and {26, 27}
Splitting t1 into {26} and {27}
Splitting t2 into {32, 33, 34} and {35, 36, 37}
Splitting t2 into {35} and {36, 37}
Splitting t2 into {36} and {37}
Splitting t3 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t3 into {32, 33, 34, 35, 18} and {19, 24, 25, 26, 27}
Splitting t3 into {24, 19} and {25, 26, 27}
Splitting t3 into {25} and {26, 27}
Splitting t3 into {26} and {27}
Splitting t4 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t4 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 29}
Splitting t4 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t4 into {26, 27} and {28, 29}
Splitting t4 into {28} and {29}
task, t1 4
task, t2 4
task, t3 1
task, t4 1
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 before t4
domain, t1 ends-by fri 1pm 5
domain, t2 ends-by wed 9am 10
domain, t3 ends-by thu 10am 10
domain, t4 ends-by mon 10am 10
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11} and {16, 17, 18, 19, 24, 25, 26, 27}
Splitting t1 into {0, 1, 2, 3} and {8, 9, 10, 11}
Splitting t1 into {0, 1} and {2, 3}
Splitting t1 into {2} and {3}
Splitting t2 into {32, 33, 8, 9, 10, 11, 16} and {17, 18, 19, 24, 25, 26, 27}
Splitting t2 into {32, 33, 8} and {16, 9, 10, 11}
Splitting t2 into {32} and {8, 33}
Splitting t2 into {8} and {33}
Splitting t3 into {12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 24} and {32, 33, 34, 35, 36, 37, 25, 26, 27, 28, 29, 30}
Splitting t3 into {12, 13, 14, 16, 17} and {18, 19, 20, 21, 22, 24}
Splitting t3 into {12, 13} and {16, 17, 14}
Splitting t3 into {12} and {13}
Splitting t4 into {13, 14, 16, 17, 18, 19, 20, 21, 22, 24, 25} and {32, 33, 34, 35, 36, 37, 26, 27, 28, 29, 30}
Splitting t4 into {13, 14, 16, 17, 18} and {19, 20, 21, 22, 24, 25}
Splitting t4 into {13, 14} and {16, 17, 18}
Splitting t4 into {13} and {14}
task, t1 4
task, t2 4
task, t3 1
task, t4 1
constraint, t1 before t2
constraint, t2 before t3
```

```
constraint, t3 before t4
domain, t1 ends-by fri 1pm 5
domain, t2 ends-by wed 9am 10
domain, t3 ends-by thu 10am 10
domain, t4 ends-by mon 10am 10
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11} and {16, 17, 18, 19, 24, 25, 26, 27}
Splitting t1 into {16, 17, 18, 19} and {24, 25, 26, 27}
Splitting t1 into {24, 25} and {26, 27}
Splitting t1 into {26} and {27}
Splitting t2 into {32} and {33}
task, t1 2
task, t2 2
task, t3 2
task, t4 1
constraint, t3 before t4
domain, t1 ends-by tue 2pm 5
domain, t2 ends-by wed 10am 20
domain, t3 ends-by fri 4pm 5
domain, t4 ends-by tue 1pm 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 3
3, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {0, 1, 2, 3, 4, 5, 8} and {9, 10, 11, 12, 13, 16, 17, 18}
Splitting t1 into {9, 10, 11, 12} and {16, 17, 18, 13}
Splitting t1 into {9, 10} and {11, 12}
Splitting t1 into {11} and {12}
Splitting t2 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 3
3, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t2 into {0, 1, 2, 3, 4, 5, 8} and {9, 10, 11, 12, 13, 16, 17, 18}
Splitting t2 into {9, 10, 11, 12} and {16, 17, 18, 13}
Splitting t2 into {16, 17} and {18, 13}
Splitting t2 into {18} and {13}
Splitting t3 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17} and {32, 33, 3
4, 35, 36, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t3 into {0, 1, 2, 3, 4, 5, 8} and {9, 10, 11, 12, 13, 16, 17}
Splitting t3 into {9, 10, 11} and {16, 17, 12, 13}
Splitting t3 into {9} and {10, 11}
Splitting t4 into {11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 24} and {32, 33, 3
4, 35, 36, 37, 38, 25, 26, 27, 28, 29, 30}
Splitting t4 into {11, 12, 13, 14, 16, 17} and {18, 19, 20, 21, 22, 24}
Splitting t4 into {11, 12, 13} and {16, 17, 14}
Splitting t4 into {11} and {12, 13}
task, t1 2
task, t2 2
task, t3 2
task, t4 1
constraint, t3 before t4
domain, t1 ends-by tue 2pm 5
domain, t2 ends-by wed 10am 20
domain, t3 ends-by fri 4pm 5
domain, t4 ends-by tue 1pm 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 3
3, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 2
9}
Splitting t1 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t1 into {26, 27} and {28, 29}
Splitting t1 into {28} and {29}
Splitting t2 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 3
3, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t2 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 2
9}
```

```

9}
Splitting t2 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t2 into {26, 27} and {28, 29}
Splitting t2 into {28} and {29}
Splitting t3 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17} and {32, 33, 3
4, 35, 36, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t3 into {32, 33, 34, 35, 36, 18, 19} and {20, 21, 24, 25, 26, 27, 28, 2
9}
Splitting t3 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t3 into {26, 27} and {28, 29}
Splitting t3 into {28} and {29}
Splitting t4 into {32, 33, 34} and {35, 36, 37, 38}
Splitting t4 into {35, 36} and {37, 38}
Splitting t4 into {37} and {38}
task, t1 4
task, t2 3
task, t3 4
task, t4 1
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 same-day t4
domain, t1 ends-by tue 2pm 5
domain, t2 ends-by mon 10am 20
domain, t3 ends-by fri 12pm 20
domain, t4 ends-by mon 12pm 20
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11} and {16, 17, 18, 19, 24, 25, 26, 27}
Splitting t1 into {0, 1, 2, 3} and {8, 9, 10, 11}
Splitting t1 into {0, 1} and {2, 3}
Splitting t1 into {0} and {1}
Splitting t2 into {32, 4, 8, 9, 10, 11, 12, 16} and {17, 18, 19, 20, 24, 25, 26,
27, 28}
Splitting t2 into {32, 8, 4, 9} and {16, 10, 11, 12}
Splitting t2 into {32, 8} and {9, 4}
Splitting t2 into {9} and {4}
Splitting t3 into {32, 33, 34, 35, 8, 9, 10, 11} and {16, 17, 18, 19, 24, 25, 26,
27}
Splitting t3 into {32, 33, 34, 35} and {8, 9, 10, 11}
Splitting t3 into {8, 9} and {10, 11}
Splitting t3 into {10} and {11}
Splitting t4 into {8, 9, 10} and {11, 12, 13, 14}
Splitting t4 into {8} and {9, 10}
task, t1 4
task, t2 3
task, t3 4
task, t4 1
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 same-day t4
domain, t1 ends-by tue 2pm 5
domain, t2 ends-by mon 10am 20
domain, t3 ends-by fri 12pm 20
domain, t4 ends-by mon 12pm 20
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11} and {16, 17, 18, 19, 24, 25, 26, 27}
Splitting t1 into {16, 17, 18, 19} and {24, 25, 26, 27}
Splitting t1 into {24, 25} and {26, 27}
Splitting t1 into {26} and {27}
Splitting t4 into {32, 33, 34} and {35, 36, 37, 38}
Splitting t4 into {35, 36} and {37, 38}
Splitting t4 into {37} and {38}
task, t1 2

```

```
task, t2 1
task, t3 2
task, t4 3
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 before t4
constraint, t3 same-day t4
domain, t1 ends-by wed 9am 10
domain, t2 ends-by thu 2pm 20
domain, t3 ends-by fri 1pm 5
domain, t4 ends-by mon 1pm 5
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13} and {16, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {0, 1, 2, 3, 4, 5} and {8, 9, 10, 11, 12, 13}
Splitting t1 into {0, 1, 2} and {3, 4, 5}
Splitting t1 into {3} and {4, 5}
Splitting t1 into {4} and {5}
Splitting t2 into {6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19} and {32, 33, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t2 into {6, 8, 9, 10, 11, 12} and {13, 14, 16, 17, 18, 19}
Splitting t2 into {8, 9, 6} and {10, 11, 12}
Splitting t2 into {8} and {9, 6}
Splitting t2 into {9} and {6}
Splitting t3 into {8, 9, 10, 11, 12, 13, 16, 17, 18, 19} and {32, 33, 34, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t3 into {8, 9, 10, 11} and {12, 13, 16, 17, 18}
Splitting t3 into {8} and {9, 10}
Splitting t4 into {10} and {11, 12}
task, t1 2
task, t2 1
task, t3 2
task, t4 3
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 before t4
constraint, t3 same-day t4
domain, t1 ends-by wed 9am 10
domain, t2 ends-by thu 2pm 20
domain, t3 ends-by fri 1pm 5
domain, t4 ends-by mon 1pm 5
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13} and {16, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {16, 17, 18, 19, 20, 21} and {24, 25, 26, 27, 28, 29}
Splitting t1 into {24, 25, 26} and {27, 28, 29}
Splitting t1 into {27} and {28, 29}
Splitting t1 into {28} and {29}
Splitting t2 into {32} and {33}
task, t1 2
task, t2 3
task, t3 2
task, t4 1
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 same-day t3
constraint, t3 before t4
domain, t1 ends-by fri 3pm 20
domain, t2 ends-by fri 1pm 20
domain, t3 ends-by mon 11am 5
domain, t4 ends-by thu 10am 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 34, 1}
```

7, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {0, 1, 2, 3, 4, 5} and {8, 9, 10, 11, 12, 13, 16}
Splitting t1 into {0} and {1, 2}
Splitting t1 into {1} and {2}
Splitting t3 into {0, 1, 2} and {3, 4, 5}
Splitting t3 into {0} and {1, 2}
Splitting t4 into {2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19} and
{32, 33, 34, 35, 36, 37, 38, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t4 into {32, 33, 34, 35, 36, 37, 38, 20} and {21, 22, 24, 25, 26, 27, 2
8, 29, 30}
Splitting t4 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t4 into {24, 25} and {21, 22}
Splitting t4 into {21} and {22}
task, t1 2
task, t2 3
task, t3 2
task, t4 1
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 same-day t3
constraint, t3 before t4
domain, t1 ends-by fri 3pm 20
domain, t2 ends-by fri 1pm 20
domain, t3 ends-by mon 11am 5
domain, t4 ends-by thu 10am 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 34, 1
7, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 34, 17, 18, 19, 20} and {21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {24} and {25, 26}
Splitting t1 into {25} and {26}
Splitting t3 into {24, 25, 26} and {27, 28, 29}
Splitting t3 into {27} and {28, 29}
Splitting t3 into {28} and {29}
Splitting t4 into {32, 33, 34} and {35, 36, 37, 38}
Splitting t4 into {35, 36} and {37, 38}
Splitting t4 into {37} and {38}
task, t1 2
task, t2 3
task, t3 2
task, t4 2
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t3 before t4
domain, t1 ends-by thu 4pm 20
domain, t2 ends-by fri 9am 10
domain, t3 ends-by mon 12pm 20
domain, t4 ends-by wed 2pm 5
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 34, 1
7, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 34, 17, 18, 19, 20} and {21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {24} and {25, 26}
Splitting t1 into {25} and {26}
Splitting t3 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17} and {32, 33, 3
4, 35, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t3 into {0, 1, 2, 3, 4, 5, 8} and {9, 10, 11, 12, 13, 16, 17}
Splitting t3 into {0, 1, 2} and {8, 3, 4, 5}
Splitting t3 into {0} and {1, 2}
Splitting t3 into {1} and {2}
Splitting t4 into {3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18, 19} and {32, 33, 3
4, 35, 36, 37, 20, 21, 24, 25, 26, 27, 28, 29}

```
Splitting t4 into {3, 4, 5, 8, 9, 10} and {11, 12, 13, 16, 17, 18, 19}
Splitting t4 into {11, 12, 13} and {16, 17, 18, 19}
Splitting t4 into {16, 17} and {18, 19}
Splitting t4 into {18} and {19}
task, t1 2
task, t2 3
task, t3 2
task, t4 2
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t3 before t4
domain, t1 ends-by thu 4pm 20
domain, t2 ends-by fri 9am 10
domain, t3 ends-by mon 12pm 20
domain, t4 ends-by wed 2pm 5
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 34, 1
7, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 34, 17, 18, 19, 20} and {21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {24} and {25, 26}
Splitting t1 into {25} and {26}
Splitting t3 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17} and {32, 33, 3
4, 35, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t3 into {32, 33, 34, 35, 18, 19, 20} and {21, 24, 25, 26, 27, 28, 29}
Splitting t3 into {24, 25, 21} and {26, 27, 28, 29}
Splitting t3 into {26, 27} and {28, 29}
Splitting t3 into {28} and {29}
Splitting t4 into {32, 33, 34} and {35, 36, 37}
Splitting t4 into {35} and {36, 37}
Splitting t4 into {36} and {37}
task, t1 1
task, t2 3
task, t3 3
task, t4 1
constraint, t1 before t2
constraint, t2 before t3
domain, t1 ends-by thu 4pm 20
domain, t2 ends-by wed 3pm 5
domain, t3 ends-by thu 12pm 5
domain, t4 ends-by thu 4pm 5
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14} and {32, 16, 1
7, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {32, 16, 17, 18, 19, 20, 21} and {22, 24, 25, 26, 27, 28, 29, 3
0}
Splitting t1 into {32, 16, 17} and {18, 19, 20, 21}
Splitting t1 into {18, 19} and {20, 21}
Splitting t1 into {18} and {19}
Splitting t2 into {32, 33, 19, 20} and {24, 25, 26, 27, 28}
Splitting t2 into {32, 33} and {19, 20}
Splitting t2 into {19} and {20}
Splitting t3 into {32, 33, 34, 35, 36} and {24, 25, 26, 27, 28}
Splitting t3 into {24, 25} and {26, 27, 28}
Splitting t3 into {24} and {25}
Splitting t4 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and
{32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t4 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 2
7, 28, 29, 30}
Splitting t4 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t4 into {26, 27} and {28, 29, 30}
Splitting t4 into {28} and {29, 30}
Splitting t4 into {29} and {30}
```

```
task, t1 1
task, t2 3
task, t3 3
task, t4 1
constraint, t1 before t2
constraint, t2 before t3
domain, t1 ends-by thu 4pm 20
domain, t2 ends-by wed 3pm 5
domain, t3 ends-by thu 12pm 5
domain, t4 ends-by thu 4pm 5
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14} and {32, 16, 1
7, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {32, 16, 17, 18, 19, 20, 21} and {22, 24, 25, 26, 27, 28, 29, 3
0}
Splitting t1 into {24, 25, 26, 22} and {27, 28, 29, 30}
Splitting t1 into {27, 28} and {29, 30}
Splitting t1 into {29} and {30}
Splitting t2 into {32} and {33}
Splitting t4 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and
{32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t4 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 2
7, 28, 29, 30}
Splitting t4 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t4 into {26, 27} and {28, 29, 30}
Splitting t4 into {28} and {29, 30}
Splitting t4 into {29} and {30}
task, t1 4
task, t2 1
task, t3 1
task, t4 3
constraint, t2 before t3
constraint, t3 before t4
domain, t1 ends-by mon 11am 10
domain, t2 ends-by tue 11am 10
domain, t3 ends-by fri 10am 10
domain, t4 ends-by thu 11am 5
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19,
24, 25, 26, 27}
Splitting t1 into {0, 1, 2, 3, 8} and {9, 10, 11, 16, 17}
Splitting t1 into {0, 1} and {8, 2, 3}
Splitting t1 into {0} and {1}
Splitting t2 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16} and {32, 3
3, 34, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t2 into {0, 1, 2, 3, 4, 5, 6} and {8, 9, 10, 11, 12, 13, 14, 16}
Splitting t2 into {8, 9, 10, 11} and {16, 12, 13, 14}
Splitting t2 into {8, 9} and {10, 11}
Splitting t2 into {8} and {9}
Splitting t3 into {10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21} and {32, 33, 34, 3
5, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t3 into {10, 11, 12, 13, 14} and {16, 17, 18, 19, 20, 21}
Splitting t3 into {16, 17, 18} and {19, 20, 21}
Splitting t3 into {19} and {20, 21}
Splitting t4 into {32, 33, 34, 35, 36} and {20, 24, 25, 26, 27, 28}
Splitting t4 into {24, 25, 20} and {26, 27, 28}
Splitting t4 into {24} and {25, 20}
Splitting t4 into {25} and {20}
task, t1 4
task, t2 1
task, t3 1
task, t4 3
```

```
constraint, t2 before t3
constraint, t3 before t4
domain, t1 ends-by mon 11am 10
domain, t2 ends-by tue 11am 10
domain, t3 ends-by fri 10am 10
domain, t4 ends-by thu 11am 5
Splitting t1 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t1 into {32, 33, 34, 35, 18} and {19, 24, 25, 26, 27}
Splitting t1 into {24, 19} and {25, 26, 27}
Splitting t1 into {25} and {26, 27}
Splitting t1 into {26} and {27}
Splitting t2 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16} and {32, 33, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t2 into {32, 33, 34, 17, 18, 19, 20, 21} and {22, 24, 25, 26, 27, 28, 29, 30}
Splitting t2 into {24, 25, 26, 22} and {27, 28, 29, 30}
Splitting t2 into {27, 28} and {29, 30}
Splitting t2 into {29} and {30}
Splitting t3 into {32, 33} and {34, 35}
Splitting t3 into {34} and {35}
task, t1 2
task, t2 3
task, t3 4
task, t4 4
task, t5 1
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 before t4
constraint, t4 before t5
domain, t1 ends-by thu 9am 20
domain, t2 ends-by tue 1pm 10
domain, t3 ends-by thu 12pm 20
domain, t4 ends-by mon 9am 5
domain, t5 ends-by wed 12pm 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10} and {11, 12, 13, 16, 17, 18, 19, 20, 21}
Splitting t1 into {0, 1, 2, 3} and {4, 5, 8, 9, 10}
Splitting t1 into {4, 5} and {8, 9, 10}
Splitting t1 into {4} and {5}
Splitting t2 into {8, 9, 10, 11, 12} and {16, 17, 18, 19, 20, 24}
Splitting t2 into {8, 9} and {10, 11, 12}
Splitting t2 into {8} and {9}
Splitting t3 into {16, 17, 18, 11} and {19, 24, 25, 26, 27}
Splitting t3 into {16, 17} and {18, 11}
Splitting t3 into {18} and {11}
Splitting t4 into {32, 33, 34, 16, 17} and {18, 19, 24, 25, 26, 27}
Splitting t4 into {32, 33} and {16, 17, 34}
Splitting t4 into {16} and {17, 34}
Splitting t5 into {32, 33, 34, 35, 36, 37, 38, 20} and {21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t5 into {32, 33, 34, 35} and {20, 36, 37, 38}
Splitting t5 into {20, 36} and {37, 38}
Splitting t5 into {20} and {36}
task, t1 2
task, t2 3
task, t3 4
task, t4 4
task, t5 1
constraint, t1 before t2
```

```
constraint, t2 before t3
constraint, t3 before t4
constraint, t4 before t5
domain, t1 ends-by thu 9am 20
domain, t2 ends-by tue 1pm 10
domain, t3 ends-by thu 12pm 20
domain, t4 ends-by mon 9am 5
domain, t5 ends-by wed 12pm 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10} and {11, 12, 13, 16, 17, 18, 19, 20, 21}
Splitting t1 into {16, 11, 12, 13} and {17, 18, 19, 20, 21}
Splitting t1 into {17, 18} and {19, 20, 21}
Splitting t1 into {19} and {20, 21}
Splitting t1 into {20} and {21}
Splitting t4 into {32} and {33, 34}
Splitting t4 into {33} and {34}
task, t1 3
task, t2 4
task, t3 1
task, t4 1
task, t5 2
constraint, t1 same-day t2
constraint, t2 before t3
domain, t1 ends-by fri 3pm 20
domain, t2 ends-by tue 2pm 5
domain, t3 ends-by thu 9am 10
domain, t4 ends-by thu 2pm 10
domain, t5 ends-by fri 2pm 5
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35, 36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t1 into {0, 1, 2, 3, 4, 8} and {9, 10, 11, 12, 16, 17}
Splitting t1 into {9, 10, 11} and {16, 17, 12}
Splitting t1 into {16} and {17, 12}
Splitting t1 into {17} and {12}
Splitting t2 into {8, 9} and {10, 11}
Splitting t2 into {8} and {9}
Splitting t3 into {13, 14, 16, 17, 18, 19, 20, 21, 22, 24, 25} and {32, 33, 34, 35, 36, 37, 38, 26, 27, 28, 29, 30}
Splitting t3 into {13, 14, 16, 17, 18} and {19, 20, 21, 22, 24, 25}
Splitting t3 into {19, 20, 21} and {24, 25, 22}
Splitting t3 into {24} and {25, 22}
Splitting t3 into {25} and {22}
Splitting t4 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t4 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t4 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t4 into {26, 27} and {28, 29, 30}
Splitting t4 into {28} and {29, 30}
Splitting t5 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t5 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 29}
Splitting t5 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t5 into {26, 27} and {28, 29}
Splitting t5 into {28} and {29}
task, t1 3
task, t2 4
task, t3 1
task, t4 1
```

```

task, t5 2
constraint, t1 same-day t2
constraint, t2 before t3
domain, t1 ends-by fri 3pm 20
domain, t2 ends-by tue 2pm 5
domain, t3 ends-by thu 9am 10
domain, t4 ends-by thu 2pm 10
domain, t5 ends-by fri 2pm 5
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35, 36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t1 into {32, 33, 34, 35, 36, 18} and {19, 20, 24, 25, 26, 27, 28}
Splitting t1 into {24, 19, 20} and {25, 26, 27, 28}
Splitting t1 into {25, 26} and {27, 28}
Splitting t1 into {27} and {28}
Splitting t2 into {24, 25} and {26, 27}
Splitting t2 into {26} and {27}
Splitting t3 into {32, 33, 34} and {35, 36, 37, 38}
Splitting t3 into {35, 36} and {37, 38}
Splitting t3 into {37} and {38}
Splitting t4 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t4 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t4 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t4 into {26, 27} and {28, 29, 30}
Splitting t4 into {28} and {29, 30}
Splitting t4 into {29} and {30}
Splitting t5 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t5 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 29}
Splitting t5 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t5 into {26, 27} and {28, 29}
Splitting t5 into {28} and {29}
task, t1 3
task, t2 1
task, t3 1
task, t4 4
task, t5 4
constraint, t1 same-day t2
constraint, t2 before t3
constraint, t2 same-day t3
constraint, t3 before t4
constraint, t4 before t5
domain, t1 ends-by fri 1pm 5
domain, t2 ends-by fri 9am 10
domain, t3 ends-by mon 9am 5
domain, t4 ends-by mon 4pm 10
domain, t5 ends-by tue 2pm 5
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12} and {16, 17, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t1 into {0, 1, 2, 3, 4} and {8, 9, 10, 11, 12}
Splitting t1 into {0, 1} and {2, 3, 4}
Splitting t1 into {2} and {3, 4}
Splitting t1 into {3} and {4}
Splitting t2 into {0, 1, 2} and {3, 4, 5}
Splitting t2 into {0} and {1, 2}
Splitting t3 into {1, 2, 3} and {4, 5, 6}
Splitting t3 into {1} and {2, 3}
Splitting t4 into {2, 3, 8, 9, 10, 11, 16} and {17, 18, 19, 24, 25, 26, 27}

```

```
Splitting t4 into {8, 2, 3} and {16, 9, 10, 11}
Splitting t4 into {8} and {2, 3}
Splitting t4 into {2} and {3}
Splitting t5 into {32, 33, 34, 35, 8, 9, 10, 11} and {16, 17, 18, 19, 24, 25, 26, 27}
Splitting t5 into {32, 33, 34, 35} and {8, 9, 10, 11}
Splitting t5 into {8, 9} and {10, 11}
Splitting t5 into {8} and {9}
task, t1 3
task, t2 1
task, t3 1
task, t4 4
task, t5 4
constraint, t1 same-day t2
constraint, t2 before t3
constraint, t2 same-day t3
constraint, t3 before t4
constraint, t4 before t5
domain, t1 ends-by fri 1pm 5
domain, t2 ends-by fri 9am 10
domain, t3 ends-by mon 9am 5
domain, t4 ends-by mon 4pm 10
domain, t5 ends-by tue 2pm 5
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12} and {16, 17, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t1 into {16, 17, 18, 19, 20} and {24, 25, 26, 27, 28}
Splitting t1 into {24, 25} and {26, 27, 28}
Splitting t1 into {26} and {27, 28}
Splitting t1 into {27} and {28}
Splitting t2 into {24} and {25}
Splitting t5 into {32, 33} and {34, 35}
Splitting t5 into {34} and {35}
task, t1 2
task, t2 4
task, t3 3
task, t4 4
task, t5 4
constraint, t1 before t2
constraint, t2 before t3
constraint, t2 same-day t3
domain, t1 ends-by tue 10am 20
domain, t2 ends-by fri 1pm 5
domain, t3 ends-by thu 11am 10
domain, t4 ends-by thu 9am 5
domain, t5 ends-by fri 12pm 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13} and {16, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {0, 1, 2, 3, 4, 5} and {8, 9, 10, 11, 12, 13}
Splitting t1 into {0, 1, 2} and {3, 4, 5}
Splitting t1 into {3} and {4, 5}
Splitting t1 into {4} and {5}
Splitting t2 into {32, 8, 9, 10, 11, 16} and {17, 18, 19, 24, 25, 26, 27}
Splitting t2 into {32, 8, 9} and {16, 10, 11}
Splitting t4 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t4 into {32, 33, 34, 35, 18} and {19, 24, 25, 26, 27}
Splitting t4 into {24, 19} and {25, 26, 27}
Splitting t4 into {24} and {19}
Splitting t5 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
```

Splitting t5 into {32, 33, 34, 35, 18} and {19, 24, 25, 26, 27}
Splitting t5 into {24, 19} and {25, 26, 27}
Splitting t5 into {25} and {26, 27}
Splitting t5 into {26} and {27}
task, t1 2
task, t2 4
task, t3 3
task, t4 4
task, t5 4
constraint, t1 before t2
constraint, t2 before t3
constraint, t2 same-day t3
domain, t1 ends-by tue 10am 20
domain, t2 ends-by fri 1pm 5
domain, t3 ends-by thu 11am 10
domain, t4 ends-by thu 9am 5
domain, t5 ends-by fri 12pm 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13} and {16, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {16, 17, 18, 19, 20, 21} and {24, 25, 26, 27, 28, 29}
Splitting t1 into {24, 25, 26} and {27, 28, 29}
Splitting t1 into {27} and {28, 29}
Splitting t1 into {28} and {29}
Splitting t4 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t4 into {32, 33, 34, 35, 18} and {19, 24, 25, 26, 27}
Splitting t4 into {24, 19} and {25, 26, 27}
Splitting t4 into {25} and {26, 27}
Splitting t4 into {26} and {27}
Splitting t5 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t5 into {32, 33, 34, 35, 18} and {19, 24, 25, 26, 27}
Splitting t5 into {24, 19} and {25, 26, 27}
Splitting t5 into {25} and {26, 27}
Splitting t5 into {26} and {27}
task, t1 3
task, t2 3
task, t3 3
task, t4 2
task, t5 4
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
constraint, t2 same-day t3
constraint, t3 same-day t4
constraint, t4 before t5
constraint, t4 same-day t5
domain, t1 ends-by wed 3pm 20
domain, t2 ends-by tue 10am 5
domain, t3 ends-by wed 1pm 5
domain, t4 ends-by tue 9am 5
domain, t5 ends-by thu 9am 5
task, t1 3
task, t2 3
task, t3 3
task, t4 2
task, t5 4
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3

```

constraint, t2 same-day t3
constraint, t3 same-day t4
constraint, t4 before t5
constraint, t4 same-day t5
domain, t1 ends-by wed 3pm 20
domain, t2 ends-by tue 10am 5
domain, t3 ends-by wed 1pm 5
domain, t4 ends-by tue 9am 5
domain, t5 ends-by thu 9am 5
task, t1 2
task, t2 2
task, t3 1
task, t4 2
task, t5 4
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
constraint, t2 same-day t3
constraint, t4 same-day t5
domain, t1 ends-by mon 10am 5
domain, t2 ends-by tue 4pm 5
domain, t3 ends-by wed 12pm 5
domain, t4 ends-by fri 4pm 10
domain, t5 ends-by fri 1pm 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 34, 1
7, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {0, 1, 2, 3, 4, 5} and {8, 9, 10, 11, 12, 13, 16}
Splitting t1 into {0} and {1, 2}
Splitting t2 into {2} and {3, 4}
Splitting t2 into {3} and {4}
Splitting t4 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 3
3, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t4 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 2
9}
Splitting t4 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t4 into {26, 27} and {28, 29}
Splitting t4 into {28} and {29}
Splitting t5 into {24, 25} and {26, 27}
Splitting t5 into {26} and {27}
task, t1 2
task, t2 2
task, t3 1
task, t4 2
task, t5 4
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
constraint, t2 same-day t3
constraint, t4 same-day t5
domain, t1 ends-by mon 10am 5
domain, t2 ends-by tue 4pm 5
domain, t3 ends-by wed 12pm 5
domain, t4 ends-by fri 4pm 10
domain, t5 ends-by fri 1pm 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 34, 1
7, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 34, 17, 18, 19, 20} and {21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {24} and {25, 26}
Splitting t1 into {25} and {26}
Splitting t4 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 3

```

```

3, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t4 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 29}
Splitting t4 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t4 into {26, 27} and {28, 29}
Splitting t4 into {28} and {29}
Splitting t5 into {24, 25} and {26, 27}
Splitting t5 into {26} and {27}
task, t1 2
task, t2 2
task, t3 1
task, t4 2
task, t5 2
constraint, t3 before t4
constraint, t3 same-day t4
domain, t1 ends-by thu 2pm 5
domain, t2 ends-by mon 2pm 5
domain, t3 ends-by tue 4pm 5
domain, t4 ends-by tue 4pm 20
domain, t5 ends-by mon 2pm 5
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t1 into {26, 27} and {28, 29}
Splitting t1 into {26} and {27}
Splitting t2 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t2 into {0, 1, 2, 3, 4, 5, 8} and {9, 10, 11, 12, 13, 16, 17, 18}
Splitting t2 into {0, 1, 2} and {8, 3, 4, 5}
Splitting t2 into {8, 3} and {4, 5}
Splitting t2 into {8} and {3}
Splitting t3 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17} and {32, 33, 34, 35, 36, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t3 into {0, 1, 2, 3, 4, 5, 6, 8} and {9, 10, 11, 12, 13, 14, 16, 17}
Splitting t3 into {9, 10, 11, 12} and {16, 17, 13, 14}
Splitting t3 into {9, 10} and {11, 12}
Splitting t3 into {11} and {12}
Splitting t5 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t5 into {0, 1, 2, 3, 4, 5, 8} and {9, 10, 11, 12, 13, 16, 17, 18}
Splitting t5 into {0, 1, 2} and {8, 3, 4, 5}
Splitting t5 into {8, 3} and {4, 5}
Splitting t5 into {8} and {3}
task, t1 2
task, t2 2
task, t3 1
task, t4 2
task, t5 2
constraint, t3 before t4
constraint, t3 same-day t4
domain, t1 ends-by thu 2pm 5
domain, t2 ends-by mon 2pm 5
domain, t3 ends-by tue 4pm 5
domain, t4 ends-by tue 4pm 20
domain, t5 ends-by mon 2pm 5
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 29}

```

```

9}
Splitting t1 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t1 into {26, 27} and {28, 29}
Splitting t1 into {28} and {29}
Splitting t2 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 3
3, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t2 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 2
9}
Splitting t2 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t2 into {26, 27} and {28, 29}
Splitting t2 into {28} and {29}
Splitting t3 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17} and {3
2, 33, 34, 35, 36, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t3 into {32, 33, 34, 35, 36, 18, 19, 20} and {21, 22, 24, 25, 26, 27, 2
8, 29, 30}
Splitting t3 into {24, 25} and {26, 27, 28}
Splitting t3 into {26} and {27, 28}
Splitting t3 into {27} and {28}
Splitting t5 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 3
3, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t5 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 2
9}
Splitting t5 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t5 into {26, 27} and {28, 29}
Splitting t5 into {28} and {29}
task, t1 2
task, t2 3
task, t3 3
task, t4 4
task, t5 1
constraint, t1 before t2
constraint, t3 same-day t4
domain, t1 ends-by fri 10am 5
domain, t2 ends-by wed 12pm 10
domain, t3 ends-by tue 12pm 10
domain, t4 ends-by thu 2pm 10
domain, t5 ends-by mon 11am 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 34, 1
7, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {0, 1, 2, 3, 4, 5} and {8, 9, 10, 11, 12, 13, 16}
Splitting t1 into {8, 9, 10} and {16, 11, 12, 13}
Splitting t1 into {16, 11} and {12, 13}
Splitting t1 into {12} and {13}
Splitting t2 into {32, 33, 34, 35, 36, 16, 17} and {18, 19, 20, 24, 25, 26, 27, 2
8}
Splitting t2 into {32, 33, 34} and {16, 17, 35, 36}
Splitting t2 into {16, 17} and {35, 36}
Splitting t2 into {16} and {17}
Splitting t3 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35,
36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t3 into {0, 1, 2, 3, 4, 8} and {9, 10, 11, 12, 16, 17}
Splitting t3 into {0, 1, 2} and {8, 3, 4}
Splitting t3 into {8} and {3, 4}
Splitting t3 into {3} and {4}
Splitting t4 into {0, 1} and {2, 3}
Splitting t4 into {2} and {3}
Splitting t5 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and
{32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t5 into {0, 1, 2, 3, 4, 5, 6, 8} and {9, 10, 11, 12, 13, 14, 16, 17, 1
8}

```

Splitting t5 into {0, 1, 2, 3} and {8, 4, 5, 6}
Splitting t5 into {0, 1} and {2, 3}
Splitting t5 into {0} and {1}
task, t1 2
task, t2 3
task, t3 3
task, t4 4
task, t5 1
constraint, t1 before t2
constraint, t3 same-day t4
domain, t1 ends-by fri 10am 5
domain, t2 ends-by wed 12pm 10
domain, t3 ends-by tue 12pm 10
domain, t4 ends-by thu 2pm 10
domain, t5 ends-by mon 11am 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 34, 1
7, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 34, 17, 18, 19, 20} and {21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {24, 25, 21} and {26, 27, 28, 29}
Splitting t1 into {26, 27} and {28, 29}
Splitting t1 into {28} and {29}
Splitting t2 into {32, 33} and {34, 35, 36}
Splitting t2 into {34} and {35, 36}
Splitting t2 into {35} and {36}
Splitting t3 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35,
36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t3 into {32, 33, 34, 35, 36, 18} and {19, 20, 24, 25, 26, 27, 28}
Splitting t3 into {24, 19, 20} and {25, 26, 27, 28}
Splitting t3 into {25, 26} and {27, 28}
Splitting t3 into {27} and {28}
Splitting t4 into {24, 25} and {26, 27}
Splitting t4 into {26} and {27}
Splitting t5 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and
{32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t5 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 2
7, 28, 29, 30}
Splitting t5 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t5 into {26, 27} and {28, 29, 30}
Splitting t5 into {28} and {29, 30}
Splitting t5 into {29} and {30}
task, t1 1
task, t2 2
task, t3 3
task, t4 2
task, t5 3
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 before t4
domain, t1 ends-by thu 12pm 10
domain, t2 ends-by tue 12pm 10
domain, t3 ends-by tue 10am 10
domain, t4 ends-by wed 10am 10
domain, t5 ends-by thu 11am 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14} and {16, 17, 1
8, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {0, 1, 2, 3, 4, 5, 6} and {8, 9, 10, 11, 12, 13, 14}
Splitting t1 into {0, 1, 2} and {3, 4, 5, 6}
Splitting t1 into {0} and {1, 2}
Splitting t1 into {1} and {2}
Splitting t2 into {2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 17, 18, 19, 20,

21, 24, 25, 26, 27, 28, 29}
Splitting t2 into {2, 3, 4, 5, 8} and {9, 10, 11, 12, 13, 16}
Splitting t2 into {2, 3} and {8, 4, 5}
Splitting t2 into {2} and {3}
Splitting t3 into {4, 8, 9, 10, 11, 12, 16, 17, 18} and {32, 33, 34, 19, 20, 24, 25, 26, 27, 28}
Splitting t3 into {8, 9, 10, 4} and {11, 12, 16, 17, 18}
Splitting t3 into {8, 9} and {10, 4}
Splitting t3 into {10} and {4}
Splitting t4 into {8, 9, 10, 11, 12, 13, 16, 17, 18, 19, 20, 21} and {32, 33, 34, 35, 36, 37, 24, 25, 26, 27, 28, 29}
Splitting t4 into {8, 9, 10, 11, 12, 13} and {16, 17, 18, 19, 20, 21}
Splitting t4 into {8, 9, 10} and {11, 12, 13}
Splitting t4 into {11} and {12, 13}
Splitting t4 into {12} and {13}
Splitting t5 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35, 36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t5 into {32, 33, 34, 35, 36, 18} and {19, 20, 24, 25, 26, 27, 28}
Splitting t5 into {24, 19, 20} and {25, 26, 27, 28}
Splitting t5 into {24} and {19, 20}
Splitting t5 into {19} and {20}
task, t1 1
task, t2 2
task, t3 3
task, t4 2
task, t5 3
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 before t4
domain, t1 ends-by thu 12pm 10
domain, t2 ends-by tue 12pm 10
domain, t3 ends-by tue 10am 10
domain, t4 ends-by wed 10am 10
domain, t5 ends-by thu 11am 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14} and {16, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {16, 17, 18, 19, 20, 21, 22} and {24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {24, 25, 26} and {27, 28, 29, 30}
Splitting t1 into {27, 28} and {29, 30}
Splitting t1 into {29} and {30}
Splitting t5 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35, 36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t5 into {32, 33, 34, 35, 36, 18} and {19, 20, 24, 25, 26, 27, 28}
Splitting t5 into {24, 19, 20} and {25, 26, 27, 28}
Splitting t5 into {25, 26} and {27, 28}
Splitting t5 into {27} and {28}
task, t1 3
task, t2 2
task, t3 3
task, t4 2
task, t5 3
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
constraint, t2 same-day t3
constraint, t4 before t5
domain, t1 ends-by thu 9am 5
domain, t2 ends-by fri 4pm 10
domain, t3 ends-by wed 4pm 10
domain, t4 ends-by fri 2pm 5

domain, t5 ends-by thu 12pm 20
Splitting t4 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 34, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
... t4 in {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} has no solution
... t4 in {32, 33, 34, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29} has no solution
task, t1 3
task, t2 2
task, t3 3
task, t4 2
task, t5 3
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
constraint, t2 same-day t3
constraint, t4 before t5
domain, t1 ends-by thu 9am 5
domain, t2 ends-by fri 4pm 10
domain, t3 ends-by wed 4pm 10
domain, t4 ends-by fri 2pm 5
domain, t5 ends-by thu 12pm 20
Splitting t4 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 34, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
... t4 in {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} has no solution
... t4 in {32, 33, 34, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29} has no solution
task, t1 2
task, t2 3
task, t3 2
task, t4 2
task, t5 3
task, t6 2
constraint, t1 before t2
constraint, t3 before t4
constraint, t4 before t5
domain, t1 ends-by fri 2pm 5
domain, t2 ends-by wed 4pm 20
domain, t3 ends-by fri 3pm 10
domain, t4 ends-by fri 10am 10
domain, t5 ends-by tue 10am 20
domain, t6 ends-by fri 12pm 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 34, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 34, 17, 18, 19, 20} and {21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 34} and {17, 18, 19, 20}
Splitting t1 into {17, 18} and {19, 20}
Splitting t1 into {17} and {18}
Splitting t2 into {32, 33, 34, 35, 36} and {20, 24, 25, 26, 27, 28}
Splitting t2 into {24, 25, 20} and {26, 27, 28}
Splitting t2 into {24} and {25, 20}
Splitting t2 into {25} and {20}
Splitting t3 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13} and {32, 33, 34, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t3 into {0, 1, 2, 3, 4, 5} and {8, 9, 10, 11, 12, 13}
Splitting t3 into {0, 1, 2} and {3, 4, 5}
Splitting t3 into {0} and {1, 2}
Splitting t4 into {2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17} and {32, 33, 34, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t4 into {2, 3, 4, 5, 8, 9} and {10, 11, 12, 13, 16, 17}
Splitting t4 into {2, 3, 4} and {8, 9, 5}

```

Splitting t4 into {2} and {3, 4}
Splitting t5 into {4, 8, 9, 10, 11, 12, 16, 17, 18, 19} and {32, 33, 34, 35, 36, 20, 24, 25, 26, 27, 28}
Splitting t5 into {4, 8, 9, 10, 11} and {12, 16, 17, 18, 19}
Splitting t5 into {8, 4} and {9, 10, 11}
Splitting t5 into {8} and {4}
Splitting t6 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t6 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 29}
Splitting t6 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t6 into {26, 27} and {28, 29}
Splitting t6 into {28} and {29}
task, t1 2
task, t2 3
task, t3 2
task, t4 2
task, t5 3
task, t6 2
constraint, t1 before t2
constraint, t3 before t4
constraint, t4 before t5
domain, t1 ends-by fri 2pm 5
domain, t2 ends-by wed 4pm 20
domain, t3 ends-by fri 3pm 10
domain, t4 ends-by fri 10am 10
domain, t5 ends-by tue 10am 20
domain, t6 ends-by fri 12pm 20
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16} and {32, 33, 34, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 34, 17, 18, 19, 20} and {21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {24, 25, 21} and {26, 27, 28, 29}
Splitting t1 into {26, 27} and {28, 29}
Splitting t1 into {28} and {29}
Splitting t2 into {32, 33} and {34, 35, 36}
Splitting t2 into {34} and {35, 36}
Splitting t2 into {35} and {36}
Splitting t3 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13} and {32, 16, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t3 into {32, 16, 17, 18, 19, 20} and {21, 24, 25, 26, 27, 28, 29}
Splitting t3 into {24, 25, 21} and {26, 27, 28, 29}
Splitting t3 into {26, 27} and {28, 29}
Splitting t3 into {28} and {29}
Splitting t4 into {32} and {33, 34}
Splitting t4 into {33} and {34}
Splitting t6 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t6 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 29}
Splitting t6 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t6 into {26, 27} and {28, 29}
Splitting t6 into {28} and {29}
task, t1 3
task, t2 2
task, t3 1
task, t4 4
task, t5 2
task, t6 3
constraint, t1 before t2
constraint, t1 same-day t2

```

```
constraint, t2 before t3
constraint, t3 before t4
constraint, t4 same-day t5
constraint, t5 before t6
domain, t1 ends-by fri 3pm 10
domain, t2 ends-by thu 1pm 10
domain, t3 ends-by fri 9am 5
domain, t4 ends-by tue 10am 20
domain, t5 ends-by wed 9am 5
domain, t6 ends-by fri 4pm 20
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10, 11} and {12, 16, 17, 18, 19, 20, 24, 25, 26}
Splitting t1 into {0, 1, 2, 3} and {8, 9, 10, 4}
Splitting t1 into {0} and {1, 2}
Splitting t1 into {1} and {2}
Splitting t2 into {4} and {5}
Splitting t3 into {6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19} and {32, 33, 34, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t3 into {6, 8, 9, 10, 11, 12} and {13, 14, 16, 17, 18, 19}
Splitting t3 into {8, 9, 6} and {10, 11, 12}
Splitting t3 into {8} and {9, 6}
Splitting t3 into {9} and {6}
Splitting t4 into {32, 33, 34, 35, 8, 9, 10, 11} and {16, 17, 18, 19, 24, 25, 26, 27}
Splitting t4 into {32, 33, 34, 35} and {8, 9, 10, 11}
Splitting t4 into {8, 9} and {10, 11}
Splitting t4 into {8} and {9}
Splitting t5 into {8, 9, 10} and {11, 12, 13}
Splitting t5 into {11} and {12, 13}
Splitting t5 into {12} and {13}
Splitting t6 into {32, 33, 34, 35, 36, 16, 17} and {18, 19, 20, 24, 25, 26, 27, 28}
Splitting t6 into {24, 18, 19, 20} and {25, 26, 27, 28}
Splitting t6 into {25, 26} and {27, 28}
Splitting t6 into {27} and {28}
task, t1 3
task, t2 2
task, t3 1
task, t4 4
task, t5 2
task, t6 3
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
constraint, t3 before t4
constraint, t4 same-day t5
constraint, t5 before t6
domain, t1 ends-by fri 3pm 10
domain, t2 ends-by thu 1pm 10
domain, t3 ends-by fri 9am 5
domain, t4 ends-by tue 10am 20
domain, t5 ends-by wed 9am 5
domain, t6 ends-by fri 4pm 20
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10, 11} and {12, 16, 17, 18, 19, 20, 24, 25, 26}
Splitting t1 into {16, 17, 18, 19} and {24, 25, 26, 20}
Splitting t1 into {24} and {25, 26}
Splitting t1 into {25} and {26}
Splitting t3 into {32} and {33, 34}
Splitting t3 into {33} and {34}
```

Splitting t5 into {32} and {33, 34}
Splitting t5 into {33} and {34}
task, t1 2
task, t2 1
task, t3 1
task, t4 3
task, t5 3
task, t6 1
constraint, t3 before t4
constraint, t3 same-day t4
constraint, t4 before t5
constraint, t4 same-day t5
domain, t1 ends-by tue 1pm 10
domain, t2 ends-by mon 2pm 5
domain, t3 ends-by mon 12pm 5
domain, t4 ends-by tue 3pm 20
domain, t5 ends-by wed 9am 10
domain, t6 ends-by fri 3pm 5
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {0, 1, 2, 3, 4, 5, 8} and {9, 10, 11, 12, 13, 16, 17, 18}
Splitting t1 into {9, 10, 11, 12} and {16, 17, 18, 13}
Splitting t1 into {9, 10} and {11, 12}
Splitting t1 into {9} and {10}
Splitting t2 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t2 into {0, 1, 2, 3, 4, 5, 6, 8} and {9, 10, 11, 12, 13, 14, 16, 17, 18}
Splitting t2 into {0, 1, 2, 3} and {8, 4, 5, 6}
Splitting t2 into {8, 4} and {5, 6}
Splitting t2 into {8} and {4}
Splitting t3 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14} and {32, 16, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t3 into {0, 1, 2, 3} and {8, 4, 5, 6}
Splitting t6 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t6 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t6 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t6 into {26, 27} and {28, 29, 30}
Splitting t6 into {28} and {29, 30}
Splitting t6 into {29} and {30}
task, t1 2
task, t2 1
task, t3 1
task, t4 3
task, t5 3
task, t6 1
constraint, t3 before t4
constraint, t3 same-day t4
constraint, t4 before t5
constraint, t4 same-day t5
domain, t1 ends-by tue 1pm 10
domain, t2 ends-by mon 2pm 5
domain, t3 ends-by mon 12pm 5
domain, t4 ends-by tue 3pm 20
domain, t5 ends-by wed 9am 10
domain, t6 ends-by fri 3pm 5
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}

Splitting t1 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t1 into {26, 27} and {28, 29}
Splitting t1 into {28} and {29}
Splitting t2 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t2 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t2 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t2 into {26, 27} and {28, 29, 30}
Splitting t2 into {28} and {29, 30}
Splitting t2 into {29} and {30}
Splitting t3 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14} and {32, 16, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t3 into {32, 16, 17, 18, 19, 20, 21} and {22, 24, 25, 26, 27, 28, 29, 30}
Splitting t6 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t6 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t6 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t6 into {26, 27} and {28, 29, 30}
Splitting t6 into {28} and {29, 30}
Splitting t6 into {29} and {30}
task, t1 3
task, t2 3
task, t3 3
task, t4 2
task, t5 4
task, t6 3
constraint, t2 same-day t3
constraint, t3 before t4
domain, t1 ends-by mon 12pm 20
domain, t2 ends-by fri 1pm 5
domain, t3 ends-by tue 1pm 10
domain, t4 ends-by mon 9am 10
domain, t5 ends-by mon 10am 5
domain, t6 ends-by fri 11am 10
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35, 36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t1 into {0, 1, 2, 3, 4, 8} and {9, 10, 11, 12, 16, 17}
Splitting t1 into {0, 1, 2} and {8, 3, 4}
Splitting t1 into {0} and {1, 2}
Splitting t2 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35, 36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t2 into {0, 1, 2, 3, 4, 8} and {9, 10, 11, 12, 16, 17}
Splitting t2 into {0, 1, 2} and {8, 3, 4}
Splitting t2 into {8} and {3, 4}
Splitting t2 into {3} and {4}
Splitting t3 into {0, 1} and {2, 3, 4}
Splitting t3 into {0} and {1}
Splitting t4 into {3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18, 19} and {32, 33, 34, 35, 36, 37, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t4 into {3, 4, 5, 8, 9, 10} and {11, 12, 13, 16, 17, 18, 19}
Splitting t4 into {3, 4, 5} and {8, 9, 10}
Splitting t4 into {3} and {4, 5}
Splitting t5 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t5 into {0, 1, 2, 3, 8} and {9, 10, 11, 16, 17}

```

Splitting t5 into {0, 1} and {8, 2, 3}
Splitting t5 into {0} and {1}
Splitting t6 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35,
36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t6 into {32, 33, 34, 35, 36, 18} and {19, 20, 24, 25, 26, 27, 28}
Splitting t6 into {24, 19, 20} and {25, 26, 27, 28}
Splitting t6 into {25, 26} and {27, 28}
Splitting t6 into {27} and {28}
task, t1 3
task, t2 3
task, t3 3
task, t4 2
task, t5 4
task, t6 3
constraint, t2 same-day t3
constraint, t3 before t4
domain, t1 ends-by mon 12pm 20
domain, t2 ends-by fri 1pm 5
domain, t3 ends-by tue 1pm 10
domain, t4 ends-by mon 9am 10
domain, t5 ends-by mon 10am 5
domain, t6 ends-by fri 11am 10
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35,
36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t1 into {32, 33, 34, 35, 36, 18} and {19, 20, 24, 25, 26, 27, 28}
Splitting t1 into {24, 19, 20} and {25, 26, 27, 28}
Splitting t1 into {25, 26} and {27, 28}
Splitting t1 into {27} and {28}
Splitting t2 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35,
36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t2 into {32, 33, 34, 35, 36, 18} and {19, 20, 24, 25, 26, 27, 28}
Splitting t2 into {24, 19, 20} and {25, 26, 27, 28}
Splitting t2 into {25, 26} and {27, 28}
Splitting t2 into {27} and {28}
Splitting t3 into {24, 25} and {26, 27, 28}
Splitting t3 into {26} and {27, 28}
Splitting t3 into {27} and {28}
Splitting t4 into {32, 33, 34} and {35, 36, 37}
Splitting t4 into {35} and {36, 37}
Splitting t4 into {36} and {37}
Splitting t5 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19,
24, 25, 26, 27}
Splitting t5 into {32, 33, 34, 35, 18} and {19, 24, 25, 26, 27}
Splitting t5 into {24, 19} and {25, 26, 27}
Splitting t5 into {25} and {26, 27}
Splitting t5 into {26} and {27}
Splitting t6 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35,
36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t6 into {32, 33, 34, 35, 36, 18} and {19, 20, 24, 25, 26, 27, 28}
Splitting t6 into {24, 19, 20} and {25, 26, 27, 28}
Splitting t6 into {25, 26} and {27, 28}
Splitting t6 into {27} and {28}
task, t1 3
task, t2 4
task, t3 4
task, t4 3
task, t5 3
task, t6 4
constraint, t1 before t2
constraint, t2 before t3

```

```
constraint, t4 before t5
constraint, t5 same-day t6
domain, t1 ends-by thu 2pm 10
domain, t2 ends-by tue 4pm 10
domain, t3 ends-by fri 2pm 5
domain, t4 ends-by thu 3pm 5
domain, t5 ends-by fri 9am 20
domain, t6 ends-by fri 11am 20
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10} and {11, 12, 16, 17, 18, 19, 20, 24}
Splitting t1 into {0, 1, 2, 3} and {8, 9, 10, 4}
Splitting t1 into {8, 9} and {10, 4}
Splitting t1 into {10} and {4}
Splitting t2 into {8, 9, 10, 11, 16, 17} and {18, 19, 24, 25, 26, 27}
Splitting t2 into {8, 9, 10} and {16, 17, 11}
Splitting t2 into {16} and {17, 11}
Splitting t2 into {17} and {11}
Splitting t3 into {32, 33, 34, 35, 16, 17} and {18, 19, 24, 25, 26, 27}
Splitting t3 into {24, 18, 19} and {25, 26, 27}
Splitting t3 into {25} and {26, 27}
Splitting t3 into {26} and {27}
Splitting t4 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16} and {32, 33, 17, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t4 into {32, 33, 17, 18, 19} and {20, 24, 25, 26, 27, 28}
Splitting t4 into {24, 25, 20} and {26, 27, 28}
Splitting t4 into {24} and {25, 20}
Splitting t4 into {25} and {20}
Splitting t5 into {32, 33, 34, 35, 36} and {24, 25, 26, 27, 28}
Splitting t5 into {24, 25} and {26, 27, 28}
Splitting t5 into {26} and {27, 28}
Splitting t5 into {27} and {28}
Splitting t6 into {24, 25} and {26, 27}
Splitting t6 into {26} and {27}
task, t1 3
task, t2 4
task, t3 4
task, t4 3
task, t5 3
task, t6 4
constraint, t1 before t2
constraint, t2 before t3
constraint, t4 before t5
constraint, t5 same-day t6
domain, t1 ends-by thu 2pm 10
domain, t2 ends-by tue 4pm 10
domain, t3 ends-by fri 2pm 5
domain, t4 ends-by thu 3pm 5
domain, t5 ends-by fri 9am 20
domain, t6 ends-by fri 11am 20
Splitting t1 into {0, 1, 2, 3, 4, 8, 9, 10} and {11, 12, 16, 17, 18, 19, 20, 24}
Splitting t1 into {16, 17, 11, 12} and {24, 18, 19, 20}
Splitting t1 into {24, 18} and {19, 20}
Splitting t1 into {19} and {20}
Splitting t2 into {24, 25} and {26, 27}
Splitting t2 into {26} and {27}
Splitting t3 into {32, 33} and {34, 35}
Splitting t3 into {34} and {35}
Splitting t4 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16} and {32, 33, 17, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t4 into {32, 33, 17, 18, 19} and {20, 24, 25, 26, 27, 28}
Splitting t4 into {24, 25, 20} and {26, 27, 28}
```

Splitting t4 into {26} and {27, 28}
Splitting t4 into {27} and {28}
Splitting t5 into {32, 33} and {34, 35, 36}
Splitting t5 into {34} and {35, 36}
Splitting t5 into {35} and {36}
Splitting t6 into {32, 33} and {34, 35}
Splitting t6 into {34} and {35}
task, t1 2
task, t2 3
task, t3 4
task, t4 4
task, t5 2
task, t6 2
constraint, t4 before t5
constraint, t5 before t6
domain, t1 ends-by wed 12pm 10
domain, t2 ends-by mon 1pm 5
domain, t3 ends-by mon 3pm 10
domain, t4 ends-by mon 1pm 10
domain, t5 ends-by tue 12pm 5
domain, t6 ends-by wed 2pm 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {0, 1, 2, 3, 4, 5, 8} and {9, 10, 11, 12, 13, 16, 17, 18}
Splitting t1 into {9, 10, 11, 12} and {16, 17, 18, 13}
Splitting t1 into {16, 17} and {18, 13}
Splitting t1 into {18} and {13}
Splitting t2 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35, 36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t2 into {0, 1, 2, 3, 4, 8} and {9, 10, 11, 12, 16, 17}
Splitting t2 into {0, 1, 2} and {8, 3, 4}
Splitting t2 into {0} and {1, 2}
Splitting t2 into {1} and {2}
Splitting t3 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19, 24, 25, 26, 27}
Splitting t3 into {0, 1, 2, 3, 8} and {9, 10, 11, 16, 17}
Splitting t3 into {0, 1} and {8, 2, 3}
Splitting t3 into {8} and {2, 3}
Splitting t3 into {2} and {3}
Splitting t4 into {0, 1, 2, 3, 8, 9, 10, 11} and {16, 17, 18, 19, 24, 25, 26, 27}
Splitting t4 into {0, 1, 2, 3} and {8, 9, 10, 11}
Splitting t4 into {0, 1} and {2, 3}
Splitting t4 into {0} and {1}
Splitting t5 into {4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18, 19} and {32, 33, 34, 35, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t5 into {4, 5, 8, 9, 10, 11} and {12, 13, 16, 17, 18, 19}
Splitting t5 into {8, 4, 5} and {9, 10, 11}
Splitting t5 into {9} and {10, 11}
Splitting t6 into {11, 12, 13, 16, 17, 18, 19, 20, 21, 24} and {32, 33, 34, 35, 36, 25, 26, 27, 28, 29}
Splitting t6 into {11, 12, 13, 16, 17} and {18, 19, 20, 21, 24}
Splitting t6 into {18, 19} and {24, 20, 21}
Splitting t6 into {18} and {19}
task, t1 2
task, t2 3
task, t3 4
task, t4 4
task, t5 2
task, t6 2
constraint, t4 before t5

```
constraint, t5 before t6
domain, t1 ends-by wed 12pm 10
domain, t2 ends-by mon 1pm 5
domain, t3 ends-by mon 3pm 10
domain, t4 ends-by mon 1pm 10
domain, t5 ends-by tue 12pm 5
domain, t6 ends-by wed 2pm 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18} and {32, 3
3, 34, 35, 36, 37, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {32, 33, 34, 35, 36, 37, 19} and {20, 21, 24, 25, 26, 27, 28, 2
9}
Splitting t1 into {24, 25, 20, 21} and {26, 27, 28, 29}
Splitting t1 into {26, 27} and {28, 29}
Splitting t1 into {28} and {29}
Splitting t2 into {0, 1, 2, 3, 4, 8, 9, 10, 11, 12, 16, 17} and {32, 33, 34, 35,
36, 18, 19, 20, 24, 25, 26, 27, 28}
Splitting t2 into {32, 33, 34, 35, 36, 18} and {19, 20, 24, 25, 26, 27, 28}
Splitting t2 into {24, 19, 20} and {25, 26, 27, 28}
Splitting t2 into {25, 26} and {27, 28}
Splitting t2 into {27} and {28}
Splitting t3 into {0, 1, 2, 3, 8, 9, 10, 11, 16, 17} and {32, 33, 34, 35, 18, 19,
24, 25, 26, 27}
Splitting t3 into {32, 33, 34, 35, 18} and {19, 24, 25, 26, 27}
Splitting t3 into {24, 19} and {25, 26, 27}
Splitting t3 into {25} and {26, 27}
Splitting t3 into {26} and {27}
Splitting t4 into {0, 1, 2, 3, 8, 9, 10, 11} and {16, 17, 18, 19, 24, 25, 26, 27}
Splitting t4 into {16, 17, 18, 19} and {24, 25, 26, 27}
Splitting t4 into {24, 25} and {26, 27}
Splitting t4 into {26} and {27}
Splitting t5 into {32, 33} and {34, 35}
Splitting t5 into {34} and {35}
task, t1 1
task, t2 1
task, t3 4
task, t4 2
task, t5 1
task, t6 1
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 before t4
constraint, t4 same-day t5
domain, t1 ends-by thu 3pm 10
domain, t2 ends-by wed 3pm 5
domain, t3 ends-by wed 3pm 5
domain, t4 ends-by tue 10am 20
domain, t5 ends-by thu 11am 10
domain, t6 ends-by tue 9am 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14} and {16, 17, 1
8, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {0, 1, 2, 3, 4, 5, 6} and {8, 9, 10, 11, 12, 13, 14}
Splitting t1 into {0, 1, 2} and {3, 4, 5, 6}
Splitting t1 into {0} and {1, 2}
Splitting t1 into {1} and {2}
Splitting t2 into {2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16} and {32, 17, 18,
19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t2 into {2, 3, 4, 5, 6, 8} and {9, 10, 11, 12, 13, 14, 16}
Splitting t2 into {2, 3, 4} and {8, 5, 6}
Splitting t2 into {2} and {3, 4}
Splitting t3 into {32, 33, 3, 8, 9, 10, 11} and {16, 17, 18, 19, 24, 25, 26, 27}
```

```
Splitting t3 into {32, 33, 3} and {8, 9, 10, 11}
Splitting t3 into {32} and {33, 3}
Splitting t3 into {33} and {3}
Splitting t4 into {8, 9, 10, 11, 12, 13, 16, 17, 18, 19, 20, 21} and {32, 33, 34, 35, 36, 37, 24, 25, 26, 27, 28, 29}
Splitting t4 into {8, 9, 10, 11, 12, 13} and {16, 17, 18, 19, 20, 21}
Splitting t4 into {8, 9, 10} and {11, 12, 13}
Splitting t4 into {8} and {9, 10}
Splitting t5 into {8, 9, 10} and {11, 12, 13, 14}
Splitting t5 into {11, 12} and {13, 14}
Splitting t5 into {13} and {14}
Splitting t6 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t6 into {0, 1, 2, 3, 4, 5, 6, 8} and {9, 10, 11, 12, 13, 14, 16, 17, 18}
Splitting t6 into {0, 1, 2, 3} and {8, 4, 5, 6}
Splitting t6 into {8, 4} and {5, 6}
Splitting t6 into {5} and {6}
task, t1 1
task, t2 1
task, t3 4
task, t4 2
task, t5 1
task, t6 1
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 before t4
constraint, t4 same-day t5
domain, t1 ends-by thu 3pm 10
domain, t2 ends-by wed 3pm 5
domain, t3 ends-by wed 3pm 5
domain, t4 ends-by tue 10am 20
domain, t5 ends-by thu 11am 10
domain, t6 ends-by tue 9am 10
Splitting t1 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14} and {16, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {16, 17, 18, 19, 20, 21, 22} and {24, 25, 26, 27, 28, 29, 30}
Splitting t1 into {24, 25, 26} and {27, 28, 29, 30}
Splitting t1 into {27, 28} and {29, 30}
Splitting t1 into {29} and {30}
Splitting t5 into {32, 33, 34} and {35, 36, 37, 38}
Splitting t5 into {35, 36} and {37, 38}
Splitting t5 into {37} and {38}
Splitting t6 into {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18} and {32, 33, 34, 35, 36, 37, 38, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t6 into {32, 33, 34, 35, 36, 37, 38, 19, 20} and {21, 22, 24, 25, 26, 27, 28, 29, 30}
Splitting t6 into {24, 25, 21, 22} and {26, 27, 28, 29, 30}
Splitting t6 into {26, 27} and {28, 29, 30}
Splitting t6 into {28} and {29, 30}
Splitting t6 into {29} and {30}
task, t1 3
task, t2 3
task, t3 3
task, t4 4
task, t5 4
task, t6 4
constraint, t1 before t2
constraint, t2 before t3
constraint, t2 same-day t3
```

```
constraint, t3 before t4
constraint, t3 same-day t4
constraint, t4 same-day t5
constraint, t5 before t6
constraint, t5 same-day t6
domain, t1 ends-by tue 12pm 20
domain, t2 ends-by mon 2pm 5
domain, t3 ends-by fri 3pm 10
domain, t4 ends-by wed 2pm 20
domain, t5 ends-by thu 9am 10
domain, t6 ends-by fri 10am 20
task, t1 3
task, t2 3
task, t3 3
task, t4 4
task, t5 4
task, t6 4
constraint, t1 before t2
constraint, t2 before t3
constraint, t2 same-day t3
constraint, t3 before t4
constraint, t3 same-day t4
constraint, t4 same-day t5
constraint, t5 before t6
constraint, t5 same-day t6
domain, t1 ends-by tue 12pm 20
domain, t2 ends-by mon 2pm 5
domain, t3 ends-by fri 3pm 10
domain, t4 ends-by wed 2pm 20
domain, t5 ends-by thu 9am 10
domain, t6 ends-by fri 10am 20
task, t1 2
task, t2 4
task, t3 3
task, t4 3
task, t5 2
task, t6 3
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
constraint, t2 same-day t3
constraint, t3 before t4
constraint, t4 before t5
constraint, t5 before t6
constraint, t5 same-day t6
domain, t1 ends-by mon 11am 20
domain, t2 ends-by thu 3pm 5
domain, t3 ends-by fri 12pm 20
domain, t4 ends-by wed 12pm 10
domain, t5 ends-by thu 4pm 5
domain, t6 ends-by wed 1pm 5
task, t1 2
task, t2 4
task, t3 3
task, t4 3
task, t5 2
task, t6 3
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
```

```

constraint, t2 same-day t3
constraint, t3 before t4
constraint, t4 before t5
constraint, t5 before t6
constraint, t5 same-day t6
domain, t1 ends-by mon 11am 20
domain, t2 ends-by thu 3pm 5
domain, t3 ends-by fri 12pm 20
domain, t4 ends-by wed 12pm 10
domain, t5 ends-by thu 4pm 5
domain, t6 ends-by wed 1pm 5
task, t1 2
task, t2 3
task, t3 2
task, t4 2
task, t5 4
task, t6 3
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 before t4
constraint, t3 same-day t4
constraint, t5 before t6
domain, t1 ends-by wed 3pm 20
domain, t2 ends-by tue 2pm 5
domain, t3 ends-by thu 1pm 5
domain, t4 ends-by thu 11am 5
domain, t5 ends-by mon 11am 5
domain, t6 ends-by thu 4pm 5
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13} and {16, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {0, 1, 2, 3, 4, 5} and {8, 9, 10, 11, 12, 13}
Splitting t1 into {8, 9, 10} and {11, 12, 13}
Splitting t1 into {8} and {9, 10}
Splitting t2 into {32, 10, 11, 12, 16, 17, 18} and {19, 20, 24, 25, 26, 27, 28}
Splitting t2 into {32, 10, 11} and {16, 17, 18, 12}
Splitting t2 into {32} and {10, 11}
Splitting t2 into {10} and {11}
Splitting t3 into {32, 33, 34, 35, 16, 17, 18, 19} and {20, 21, 24, 25, 26, 27, 28}
Splitting t3 into {32, 33, 34, 35} and {16, 17, 18, 19}
Splitting t3 into {16, 17} and {18, 19}
Splitting t3 into {18} and {19}
Splitting t5 into {0, 1, 2, 3, 32, 8, 9, 10} and {11, 16, 17, 18, 19, 24, 25, 26, 27}
Splitting t5 into {0, 1, 2, 3} and {32, 8, 10, 9}
Splitting t5 into {0, 1} and {2, 3}
Splitting t5 into {0} and {1}
Splitting t6 into {4, 8, 9, 10, 11, 12, 16, 17, 18, 19} and {32, 33, 34, 35, 36, 20, 24, 25, 26, 27, 28}
Splitting t6 into {32, 33, 34, 35, 36} and {20, 24, 25, 26, 27, 28}
Splitting t6 into {24, 25, 20} and {26, 27, 28}
Splitting t6 into {26} and {27, 28}
Splitting t6 into {27} and {28}
task, t1 2
task, t2 3
task, t3 2
task, t4 2
task, t5 4
task, t6 3
constraint, t1 before t2

```

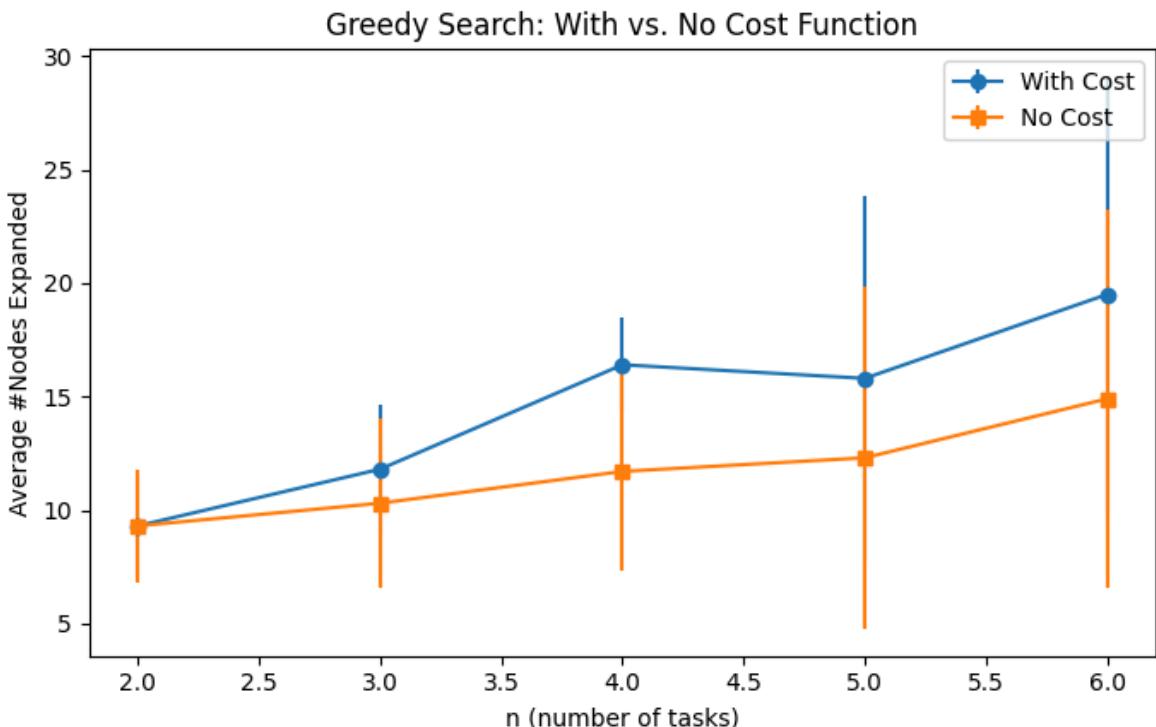
```

constraint, t2 before t3
constraint, t3 before t4
constraint, t3 same-day t4
constraint, t5 before t6
domain, t1 ends-by wed 3pm 20
domain, t2 ends-by tue 2pm 5
domain, t3 ends-by thu 1pm 5
domain, t4 ends-by thu 11am 5
domain, t5 ends-by mon 11am 5
domain, t6 ends-by thu 4pm 5
Splitting t1 into {0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13} and {16, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29}
Splitting t1 into {16, 17, 18, 19, 20, 21} and {24, 25, 26, 27, 28, 29}
Splitting t1 into {24, 25, 26} and {27, 28, 29}
Splitting t1 into {27} and {28, 29}
Splitting t1 into {28} and {29}
Splitting t5 into {0, 1, 2, 3, 32, 8, 9, 10} and {11, 16, 17, 18, 19, 24, 25, 26, 27}
Splitting t5 into {16, 17, 18, 11} and {19, 24, 25, 26, 27}
Splitting t5 into {24, 19} and {25, 26, 27}
Splitting t5 into {25} and {26, 27}
Splitting t5 into {26} and {27}
Splitting t6 into {32, 33} and {34, 35, 36}
Splitting t6 into {34} and {35, 36}
Splitting t6 into {35} and {36}

```

==== Average Nodes Expanded (TRIALS = 10) ===

n	with_cost	no_cost	gain(%)
2	9.30	9.30	0.0%
3	11.80	10.30	-14.6%
4	16.40	11.70	-40.2%
5	15.80	12.30	-28.5%
6	19.50	14.90	-30.9%



Question 4 (5 marks)

Compare the Depth-First Search (DFS) solver to the Depth-First Search solver using forward checking with Minimum Remaining Values heuristic (DFS-MRV). For this question, ignore the costs associated with the CSP problems.

- What is the worst case time and space complexity of each algorithm (give a general form in terms of fuzzy scheduling problem sizes)? (1 mark)
- What are the properties of the search algorithms (completeness, optimality)? (1 mark)
- Give an example of a problem that is *easier* for the DFS-MRV solver than it is for the DFS solver, and explain why (1 mark)
- Empirically compare the quality of the first solution found by DFS and DFS-MRV compared to the optimal solution (1 mark)
- Empirically compare DFS-MRV with DFS in terms of the number of nodes expanded (1 mark)

For the empirical evaluations, run the two algorithms on a variety of problems of size n for varying n . Note that the domain splitting CSP solver with costs should always find an optimal solution.

```
In [24]: # Running this code will show you the difference between dfs_mrv and dfs in term
import matplotlib.pyplot as plt
import random
from statistics import mean

def main():
    Ns = [2, 3, 4, 5, 6]      # Different problem sizes
    TRIALS = 5                # Number of trials for each n
    BASE_SEED = 42             # Base random seed (for reproducibility)
    MAX_ATTEMPTS = 5          # Max retries if a generated problem is unsolvable

    dfs_results = []
    mrv_results = []

    global display, num_expanded
    display = False # Disable detailed DSL printing

    for n in Ns:
        dfs_counts = []
        mrv_counts = []

        print("\n====")
        print(f" Problem size n = {n}")
        print("====")

        for trial in range(TRIALS):
            # Set a unique random seed for each trial to ensure reproducibility
            seed = BASE_SEED + n * 100 + trial
            random.seed(seed)

            # Retry multiple times if an unsolvable CSP is generated
            for attempt in range(MAX_ATTEMPTS):
                spec = generate_problem(n)
                csp1 = create_CSP_from_spec(spec)
                csp2 = create_CSP_from_spec(spec)
```

```

# ---- Plain DFS ----
num_expanded = 0
sol1 = dfs_solve1(csp1)
dfs_nodes = num_expanded

# ---- MRV + Forward Checking ----
num_expanded = 0
sol2 = mrv_dfs_solve1(csp2)
mrv_nodes = num_expanded

# Only record results if both solvers found a solution
if sol1 is not None and sol2 is not None:
    dfs_counts.append(dfs_nodes)
    mrv_counts.append(mrv_nodes)

# Print the "First Solution" content of the first trial
if trial == 0:
    print(f"--- DFS (first solution) ---")
    print(f"Nodes expanded: {dfs_nodes}")
    print("Solution:", sol1)

    print(f"--- MRV + Forward Checking (first solution) ---")
    print(f"Nodes expanded: {mrv_nodes}")
    print("Solution:", sol2)

    break # Stop retry loop if successful
else:
    print(f" n={n}, trial={trial+1}, attempt {attempt+1}: Unsolv

# Compute average number of nodes expanded for this n
avg_dfs = mean(dfs_counts) if dfs_counts else float("nan")
avg_mrv = mean(mrv_counts) if mrv_counts else float("nan")

dfs_results.append(avg_dfs)
mrv_results.append(avg_mrv)

print(f" n = {n} | DFS avg expanded nodes: {avg_dfs:.2f} | MRV avg expanded nodes: {avg_mrv:.2f}")

# ----- Plot results -----
plt.plot(Ns, dfs_results, marker='o', label='DFS')
plt.plot(Ns, mrv_results, marker='s', label='DFS + MRV + Forward Checking')
plt.xlabel("n (problem size)")
plt.ylabel("Average nodes expanded")
plt.title("DFS vs DFS-MRV (ignoring cost)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

if __name__ == "__main__":
    main()

```

```
=====
Problem size n = 2
=====
task, t1 3
task, t2 1
constraint, t1 before t2
domain, t1 ends-by fri 4pm 20
domain, t2 ends-by tue 9am 5
task, t1 3
task, t2 1
constraint, t1 before t2
domain, t1 ends-by fri 4pm 20
domain, t2 ends-by tue 9am 5
Nodes expanded to reach solution: 5
Nodes expanded to reach solution: 2
--- DFS (first solution) ---
Nodes expanded: 5
Solution: {'t1': 0, 't2': 3}
--- MRV + Forward Checking (first solution) ---
Nodes expanded: 2
Solution: {'t1': 0, 't2': 3}
task, t1 3
task, t2 4
constraint, t1 before t2
domain, t1 ends-by tue 1pm 5
domain, t2 ends-by mon 10am 10
task, t1 3
task, t2 4
constraint, t1 before t2
domain, t1 ends-by tue 1pm 5
domain, t2 ends-by mon 10am 10
Nodes expanded to reach solution: 5
Nodes expanded to reach solution: 5
task, t1 4
task, t2 4
domain, t1 ends-by wed 10am 20
domain, t2 ends-by tue 2pm 10
task, t1 4
task, t2 4
domain, t1 ends-by wed 10am 20
domain, t2 ends-by tue 2pm 10
Nodes expanded to reach solution: 2
Nodes expanded to reach solution: 2
task, t1 2
task, t2 3
constraint, t1 before t2
domain, t1 ends-by fri 4pm 5
domain, t2 ends-by tue 1pm 5
task, t1 2
task, t2 3
constraint, t1 before t2
domain, t1 ends-by fri 4pm 5
domain, t2 ends-by tue 1pm 5
Nodes expanded to reach solution: 4
Nodes expanded to reach solution: 4
task, t1 1
task, t2 4
constraint, t1 same-day t2
domain, t1 ends-by thu 11am 20
domain, t2 ends-by tue 2pm 20
```

```

task, t1 1
task, t2 4
constraint, t1 same-day t2
domain, t1 ends-by thu 11am 20
domain, t2 ends-by tue 2pm 20
Nodes expanded to reach solution: 2
Nodes expanded to reach solution: 2
n = 2 | DFS avg expanded nodes: 3.60 | MRV avg expanded nodes: 3.00

=====
Problem size n = 3
=====
task, t1 1
task, t2 2
task, t3 1
constraint, t1 before t2
domain, t1 ends-by fri 1pm 20
domain, t2 ends-by mon 2pm 10
domain, t3 ends-by mon 11am 20
task, t1 1
task, t2 2
task, t3 1
constraint, t1 before t2
domain, t1 ends-by fri 1pm 20
domain, t2 ends-by mon 2pm 10
domain, t3 ends-by mon 11am 20
Nodes expanded to reach solution: 4
Nodes expanded to reach solution: 4
--- DFS (first solution) ---
Nodes expanded: 4
Solution: {'t1': 0, 't2': 1, 't3': 0}
--- MRV + Forward Checking (first solution) ---
Nodes expanded: 4
Solution: {'t2': 1, 't1': 0, 't3': 0}
task, t1 2
task, t2 4
task, t3 2
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
domain, t1 ends-by wed 1pm 5
domain, t2 ends-by tue 11am 5
domain, t3 ends-by wed 10am 10
task, t1 2
task, t2 4
task, t3 2
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
domain, t1 ends-by wed 1pm 5
domain, t2 ends-by tue 11am 5
domain, t3 ends-by wed 10am 10
Nodes expanded to reach solution: 11
Nodes expanded to reach solution: 5
task, t1 4
task, t2 4
task, t3 1
constraint, t1 before t2
constraint, t2 before t3
domain, t1 ends-by fri 2pm 10

```

```
domain, t2 ends-by thu 3pm 20
domain, t3 ends-by wed 2pm 20
task, t1 4
task, t2 4
task, t3 1
constraint, t1 before t2
constraint, t2 before t3
domain, t1 ends-by fri 2pm 10
domain, t2 ends-by thu 3pm 20
domain, t3 ends-by wed 2pm 20
Nodes expanded to reach solution: 18
Nodes expanded to reach solution: 3
task, t1 3
task, t2 1
task, t3 2
constraint, t1 same-day t2
constraint, t2 before t3
domain, t1 ends-by tue 1pm 10
domain, t2 ends-by wed 9am 20
domain, t3 ends-by wed 11am 5
task, t1 3
task, t2 1
task, t3 2
constraint, t1 same-day t2
constraint, t2 before t3
domain, t1 ends-by tue 1pm 10
domain, t2 ends-by wed 9am 20
domain, t3 ends-by wed 11am 5
Nodes expanded to reach solution: 4
Nodes expanded to reach solution: 3
task, t1 2
task, t2 1
task, t3 3
constraint, t1 before t2
constraint, t2 before t3
constraint, t2 same-day t3
domain, t1 ends-by mon 10am 5
domain, t2 ends-by thu 10am 10
domain, t3 ends-by wed 9am 5
task, t1 2
task, t2 1
task, t3 3
constraint, t1 before t2
constraint, t2 before t3
constraint, t2 same-day t3
domain, t1 ends-by mon 10am 5
domain, t2 ends-by thu 10am 10
domain, t3 ends-by wed 9am 5
Nodes expanded to reach solution: 8
Nodes expanded to reach solution: 11
n = 3 | DFS avg expanded nodes: 9.00 | MRV avg expanded nodes: 5.20
```

```
=====
Problem size n = 4
=====
task, t1 3
task, t2 4
task, t3 4
task, t4 4
constraint, t1 before t2
```

```
constraint, t3 same-day t4
domain, t1 ends-by thu 10am 5
domain, t2 ends-by wed 12pm 5
domain, t3 ends-by thu 3pm 20
domain, t4 ends-by tue 3pm 10
task, t1 3
task, t2 4
task, t3 4
task, t4 4
constraint, t1 before t2
constraint, t3 same-day t4
domain, t1 ends-by thu 10am 5
domain, t2 ends-by wed 12pm 5
domain, t3 ends-by thu 3pm 20
domain, t4 ends-by tue 3pm 10
Nodes expanded to reach solution: 7
Nodes expanded to reach solution: 7
--- DFS (first solution) ---
Nodes expanded: 7
Solution: {'t1': 0, 't2': 3, 't3': 0, 't4': 0}
--- MRV + Forward Checking (first solution) ---
Nodes expanded: 7
Solution: {'t2': 3, 't1': 0, 't3': 0, 't4': 0}
task, t1 2
task, t2 1
task, t3 3
task, t4 3
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
constraint, t3 before t4
domain, t1 ends-by fri 10am 10
domain, t2 ends-by wed 4pm 20
domain, t3 ends-by tue 10am 10
domain, t4 ends-by thu 4pm 5
task, t1 2
task, t2 1
task, t3 3
task, t4 3
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
constraint, t3 before t4
domain, t1 ends-by fri 10am 10
domain, t2 ends-by wed 4pm 20
domain, t3 ends-by tue 10am 10
domain, t4 ends-by thu 4pm 5
Nodes expanded to reach solution: 14
Nodes expanded to reach solution: 12
task, t1 3
task, t2 3
task, t3 1
task, t4 3
constraint, t1 before t2
constraint, t3 before t4
domain, t1 ends-by mon 3pm 20
domain, t2 ends-by tue 4pm 20
domain, t3 ends-by mon 10am 5
domain, t4 ends-by mon 9am 10
task, t1 3
```

```
task, t2 3
task, t3 1
task, t4 3
constraint, t1 before t2
constraint, t3 before t4
domain, t1 ends-by mon 3pm 20
domain, t2 ends-by tue 4pm 20
domain, t3 ends-by mon 10am 5
domain, t4 ends-by mon 9am 10
Nodes expanded to reach solution: 8
Nodes expanded to reach solution: 5
task, t1 4
task, t2 2
task, t3 1
task, t4 4
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 same-day t4
domain, t1 ends-by thu 3pm 10
domain, t2 ends-by wed 2pm 10
domain, t3 ends-by tue 9am 20
domain, t4 ends-by wed 4pm 10
task, t1 4
task, t2 2
task, t3 1
task, t4 4
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 same-day t4
domain, t1 ends-by thu 3pm 10
domain, t2 ends-by wed 2pm 10
domain, t3 ends-by tue 9am 20
domain, t4 ends-by wed 4pm 10
Nodes expanded to reach solution: 14
Nodes expanded to reach solution: 10
task, t1 4
task, t2 2
task, t3 2
task, t4 2
constraint, t2 same-day t3
constraint, t3 before t4
constraint, t3 same-day t4
domain, t1 ends-by tue 11am 20
domain, t2 ends-by mon 3pm 10
domain, t3 ends-by mon 12pm 10
domain, t4 ends-by thu 9am 10
task, t1 4
task, t2 2
task, t3 2
task, t4 2
constraint, t2 same-day t3
constraint, t3 before t4
constraint, t3 same-day t4
domain, t1 ends-by tue 11am 20
domain, t2 ends-by mon 3pm 10
domain, t3 ends-by mon 12pm 10
domain, t4 ends-by thu 9am 10
Nodes expanded to reach solution: 6
Nodes expanded to reach solution: 4
n = 4 | DFS avg expanded nodes: 9.80 | MRV avg expanded nodes: 7.60
```

```
=====
Problem size n = 5
=====
task, t1 4
task, t2 1
task, t3 2
task, t4 4
task, t5 2
constraint, t1 before t2
constraint, t2 before t3
constraint, t2 same-day t3
domain, t1 ends-by mon 11am 5
domain, t2 ends-by fri 9am 10
domain, t3 ends-by thu 2pm 10
domain, t4 ends-by thu 3pm 20
domain, t5 ends-by thu 10am 5
task, t1 4
task, t2 1
task, t3 2
task, t4 4
task, t5 2
constraint, t1 before t2
constraint, t2 before t3
constraint, t2 same-day t3
domain, t1 ends-by mon 11am 5
domain, t2 ends-by fri 9am 10
domain, t3 ends-by thu 2pm 10
domain, t4 ends-by thu 3pm 20
domain, t5 ends-by thu 10am 5
Nodes expanded to reach solution: 14
Nodes expanded to reach solution: 10
--- DFS (first solution) ---
Nodes expanded: 14
Solution: {'t1': 0, 't2': 4, 't3': 5, 't4': 0, 't5': 0}
--- MRV + Forward Checking (first solution) ---
Nodes expanded: 10
Solution: {'t1': 0, 't4': 0, 't3': 5, 't2': 4, 't5': 0}
task, t1 2
task, t2 1
task, t3 4
task, t4 2
task, t5 3
constraint, t1 before t2
constraint, t1 same-day t2
domain, t1 ends-by wed 10am 5
domain, t2 ends-by fri 1pm 5
domain, t3 ends-by fri 4pm 10
domain, t4 ends-by fri 11am 10
domain, t5 ends-by fri 3pm 20
task, t1 2
task, t2 1
task, t3 4
task, t4 2
task, t5 3
constraint, t1 before t2
constraint, t1 same-day t2
domain, t1 ends-by wed 10am 5
domain, t2 ends-by fri 1pm 5
domain, t3 ends-by fri 4pm 10
```

```
domain, t4 ends-by fri 11am 10
domain, t5 ends-by fri 3pm 20
Nodes expanded to reach solution: 7
Nodes expanded to reach solution: 5
task, t1 3
task, t2 1
task, t3 3
task, t4 3
task, t5 1
constraint, t1 same-day t2
constraint, t2 before t3
constraint, t3 before t4
constraint, t4 before t5
constraint, t4 same-day t5
domain, t1 ends-by mon 11am 10
domain, t2 ends-by thu 10am 10
domain, t3 ends-by fri 9am 10
domain, t4 ends-by thu 10am 20
domain, t5 ends-by tue 2pm 10
task, t1 3
task, t2 1
task, t3 3
task, t4 3
task, t5 1
constraint, t1 same-day t2
constraint, t2 before t3
constraint, t3 before t4
constraint, t4 before t5
constraint, t4 same-day t5
domain, t1 ends-by mon 11am 10
domain, t2 ends-by thu 10am 10
domain, t3 ends-by fri 9am 10
domain, t4 ends-by thu 10am 20
domain, t5 ends-by tue 2pm 10
Nodes expanded to reach solution: 56
Nodes expanded to reach solution: 6
task, t1 2
task, t2 1
task, t3 2
task, t4 4
task, t5 1
constraint, t1 before t2
constraint, t2 before t3
constraint, t2 same-day t3
constraint, t3 before t4
domain, t1 ends-by mon 11am 10
domain, t2 ends-by mon 3pm 10
domain, t3 ends-by mon 1pm 20
domain, t4 ends-by mon 11am 5
domain, t5 ends-by tue 12pm 10
task, t1 2
task, t2 1
task, t3 2
task, t4 4
task, t5 1
constraint, t1 before t2
constraint, t2 before t3
constraint, t2 same-day t3
constraint, t3 before t4
domain, t1 ends-by mon 11am 10
```

```
domain, t2 ends-by mon 3pm 10
domain, t3 ends-by mon 1pm 20
domain, t4 ends-by mon 11am 5
domain, t5 ends-by tue 12pm 10
Nodes expanded to reach solution: 14
Nodes expanded to reach solution: 21
task, t1 1
task, t2 3
task, t3 4
task, t4 4
task, t5 1
constraint, t2 before t3
constraint, t2 same-day t3
constraint, t4 before t5
constraint, t4 same-day t5
domain, t1 ends-by mon 12pm 10
domain, t2 ends-by tue 1pm 20
domain, t3 ends-by tue 9am 20
domain, t4 ends-by thu 12pm 5
domain, t5 ends-by wed 12pm 10
task, t1 1
task, t2 3
task, t3 4
task, t4 4
task, t5 1
constraint, t2 before t3
constraint, t2 same-day t3
constraint, t4 before t5
constraint, t4 same-day t5
domain, t1 ends-by mon 12pm 10
domain, t2 ends-by tue 1pm 20
domain, t3 ends-by tue 9am 20
domain, t4 ends-by thu 12pm 5
domain, t5 ends-by wed 12pm 10
Nodes expanded to reach solution: 12
Nodes expanded to reach solution: 8
n = 5 | DFS avg expanded nodes: 20.60 | MRV avg expanded nodes: 10.00
```

```
=====
Problem size n = 6
=====
task, t1 3
task, t2 4
task, t3 4
task, t4 1
task, t5 3
task, t6 4
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 before t4
constraint, t3 same-day t4
constraint, t4 before t5
constraint, t4 same-day t5
constraint, t5 same-day t6
domain, t1 ends-by thu 4pm 5
domain, t2 ends-by thu 11am 5
domain, t3 ends-by wed 4pm 5
domain, t4 ends-by tue 4pm 10
domain, t5 ends-by thu 9am 5
domain, t6 ends-by tue 12pm 20
```

```
task, t1 3
task, t2 4
task, t3 4
task, t4 1
task, t5 3
task, t6 4
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 before t4
constraint, t3 same-day t4
constraint, t4 before t5
constraint, t4 same-day t5
constraint, t5 same-day t6
domain, t1 ends-by thu 4pm 5
domain, t2 ends-by thu 11am 5
domain, t3 ends-by wed 4pm 5
domain, t4 ends-by tue 4pm 10
domain, t5 ends-by thu 9am 5
domain, t6 ends-by tue 12pm 20
n=6, trial=1, attempt 1: Unsolvable problem, retrying...
task, t1 4
task, t2 2
task, t3 2
task, t4 3
task, t5 1
task, t6 1
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
constraint, t3 before t4
constraint, t5 before t6
domain, t1 ends-by mon 9am 20
domain, t2 ends-by mon 4pm 10
domain, t3 ends-by tue 9am 20
domain, t4 ends-by fri 11am 5
domain, t5 ends-by fri 4pm 5
domain, t6 ends-by wed 2pm 10
task, t1 4
task, t2 2
task, t3 2
task, t4 3
task, t5 1
task, t6 1
constraint, t1 before t2
constraint, t1 same-day t2
constraint, t2 before t3
constraint, t3 before t4
constraint, t5 before t6
domain, t1 ends-by mon 9am 20
domain, t2 ends-by mon 4pm 10
domain, t3 ends-by tue 9am 20
domain, t4 ends-by fri 11am 5
domain, t5 ends-by fri 4pm 5
domain, t6 ends-by wed 2pm 10
Nodes expanded to reach solution: 24
Nodes expanded to reach solution: 6
--- DFS (first solution) ---
Nodes expanded: 24
Solution: {'t1': 0, 't2': 4, 't3': 8, 't4': 10, 't5': 0, 't6': 1}
--- MRV + Forward Checking (first solution) ---
```

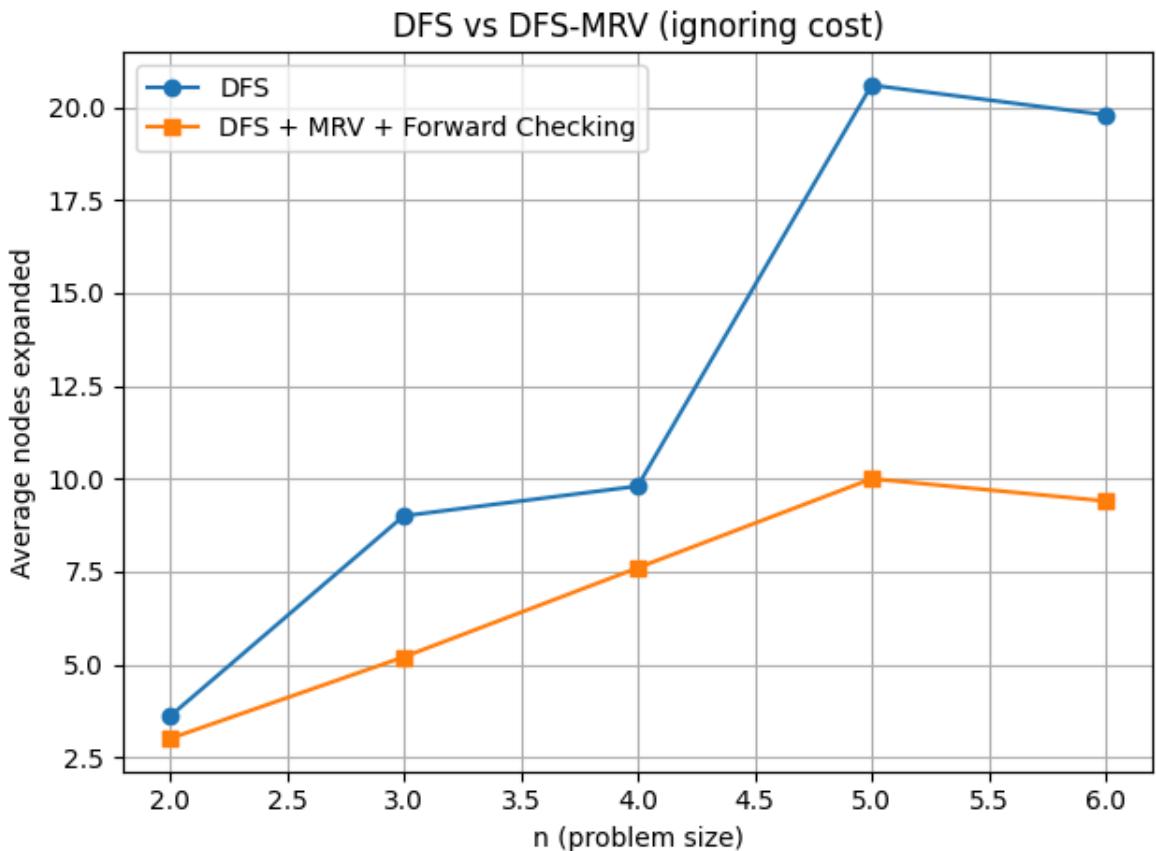
Nodes expanded: 6
Solution: {'t1': 0, 't2': 4, 't3': 8, 't4': 10, 't5': 0, 't6': 1}
task, t1 4
task, t2 3
task, t3 2
task, t4 1
task, t5 3
task, t6 2
constraint, t1 before t2
constraint, t2 before t3
constraint, t5 before t6
domain, t1 ends-by mon 11am 10
domain, t2 ends-by tue 1pm 20
domain, t3 ends-by mon 12pm 20
domain, t4 ends-by tue 10am 5
domain, t5 ends-by fri 12pm 10
domain, t6 ends-by tue 4pm 10
task, t1 4
task, t2 3
task, t3 2
task, t4 1
task, t5 3
task, t6 2
constraint, t1 before t2
constraint, t2 before t3
constraint, t5 before t6
domain, t1 ends-by mon 11am 10
domain, t2 ends-by tue 1pm 20
domain, t3 ends-by mon 12pm 20
domain, t4 ends-by tue 10am 5
domain, t5 ends-by fri 12pm 10
domain, t6 ends-by tue 4pm 10
Nodes expanded to reach solution: 19
Nodes expanded to reach solution: 6
task, t1 4
task, t2 3
task, t3 4
task, t4 1
task, t5 4
task, t6 2
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 before t4
constraint, t5 same-day t6
domain, t1 ends-by fri 2pm 20
domain, t2 ends-by mon 10am 10
domain, t3 ends-by tue 12pm 5
domain, t4 ends-by thu 4pm 5
domain, t5 ends-by thu 9am 20
domain, t6 ends-by mon 9am 5
task, t1 4
task, t2 3
task, t3 4
task, t4 1
task, t5 4
task, t6 2
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 before t4
constraint, t5 same-day t6

```
domain, t1 ends-by fri 2pm 20
domain, t2 ends-by mon 10am 10
domain, t3 ends-by tue 12pm 5
domain, t4 ends-by thu 4pm 5
domain, t5 ends-by thu 9am 20
domain, t6 ends-by mon 9am 5
Nodes expanded to reach solution: 25
Nodes expanded to reach solution: 10
task, t1 2
task, t2 2
task, t3 4
task, t4 1
task, t5 3
task, t6 1
constraint, t1 before t2
constraint, t2 same-day t3
constraint, t4 before t5
constraint, t4 same-day t5
constraint, t5 before t6
constraint, t5 same-day t6
domain, t1 ends-by mon 11am 20
domain, t2 ends-by mon 2pm 20
domain, t3 ends-by wed 4pm 10
domain, t4 ends-by mon 11am 10
domain, t5 ends-by thu 1pm 5
domain, t6 ends-by fri 2pm 20
task, t1 2
task, t2 2
task, t3 4
task, t4 1
task, t5 3
task, t6 1
constraint, t1 before t2
constraint, t2 same-day t3
constraint, t4 before t5
constraint, t4 same-day t5
constraint, t5 before t6
constraint, t5 same-day t6
domain, t1 ends-by mon 11am 20
domain, t2 ends-by mon 2pm 20
domain, t3 ends-by wed 4pm 10
domain, t4 ends-by mon 11am 10
domain, t5 ends-by thu 1pm 5
domain, t6 ends-by fri 2pm 20
Nodes expanded to reach solution: 13
Nodes expanded to reach solution: 9
task, t1 3
task, t2 2
task, t3 4
task, t4 3
task, t5 4
task, t6 2
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 same-day t4
domain, t1 ends-by thu 2pm 10
domain, t2 ends-by mon 2pm 5
domain, t3 ends-by thu 2pm 10
domain, t4 ends-by thu 9am 10
domain, t5 ends-by wed 12pm 5
```

```

domain, t6 ends-by mon 4pm 5
task, t1 3
task, t2 2
task, t3 4
task, t4 3
task, t5 4
task, t6 2
constraint, t1 before t2
constraint, t2 before t3
constraint, t3 same-day t4
domain, t1 ends-by thu 2pm 10
domain, t2 ends-by mon 2pm 5
domain, t3 ends-by thu 2pm 10
domain, t4 ends-by thu 9am 10
domain, t5 ends-by wed 12pm 5
domain, t6 ends-by mon 4pm 5
Nodes expanded to reach solution: 18
Nodes expanded to reach solution: 16
n = 6 | DFS avg expanded nodes: 19.80 | MRV avg expanded nodes: 9.40

```



Answers for Question 4

1 Time and Space Complexity

For a fuzzy scheduling CSP with **(n)** tasks (variables), each having a domain size of up to **(d)** possible start times, the worst-case time complexity of a standard depth-first search (DFS) solver is:

$$O(d^n)$$

This is because the solver may need to explore all possible assignments across all variables.

The space complexity for DFS is:

$$O(n)$$

since it only needs to store the current assignment path along the recursion stack.

DFS + MRV + Forward Checking

When the solver incorporates the **Minimum Remaining Values (MRV)** heuristic and **forward checking**, the worst-case time complexity remains:

$$O(d^n)$$

because the algorithm may still explore the entire search tree. However, each node expansion introduces additional overhead of approximately $O(e)$, where **(e)** is the number of constraints, due to domain pruning and constraint propagation.

The space complexity increases to:

$$O(n \cdot (d + e))$$

because the solver must maintain the current domain states and constraint consistency information during the search.

Explanation of Parameters

Symbol	Meaning in fuzzy scheduling CSP	Corresponding code variable
(n)	Number of tasks (variables)	<code>len(csp.domains)</code>
(d)	Maximum domain size (possible start times per task)	<code>max(len(dom) for dom in node.domains.values())</code>
(e)	Number of constraints (e.g., before/same-day constraints)	<code>len(csp.constraints)</code>

These parameters capture the scale of the problem and directly influence the performance of the search algorithms.

2 Properties of the Search Algorithms

Both the standard DFS solver and the DFS-MRV solver are **complete** for finite-domain fuzzy scheduling CSPs — they will eventually find a solution if one exists. However, neither approach guarantees **optimality** when costs are ignored, as they return the first feasible solution they find. If cost-based reasoning is incorporated (e.g., using a cost function as in Question 3), then the search can be made optimal with respect to total scheduling cost.

3 Example Where DFS-MRV Performs Better

Consider a fuzzy scheduling problem where one task is highly constrained, for example, it must start on **Monday at 9 a.m.**, while other tasks have many possible time slots. In this case, the MRV heuristic quickly selects the most constrained variable first, assigns it early, and uses forward checking to prune inconsistent values for the remaining tasks. This leads to earlier detection of conflicts and significantly reduces the search space. A plain DFS solver, by contrast, might assign variables with large domains first, exploring many dead-end branches before eventually finding a solution.

4 Comparison of the First Solution Found by DFS and DFS-MRV (Ignoring Cost)

In this question, we ignore the cost function and focus only on finding valid solutions that satisfy all hard constraints. Under this setting, any feasible solution is considered optimal. The *first solution* refers to the first valid assignment each algorithm finds during the search.

We ran both the standard DFS solver and the DFS-MRV solver (with forward checking) on randomly generated fuzzy scheduling CSPs for problem sizes $n = 2-5$. For each case, we recorded the number of nodes expanded and the first solution found.

n	DFS nodes expanded	MRV nodes expanded	DFS first solution	MRV first solution
2	5	2	{'t1': 0, 't2': 3}	{'t1': 0, 't2': 3}
3	4	4	{'t1': 0, 't2': 1, 't3': 0}	{'t2': 1, 't1': 0, 't3': 0}
4	7	7	{'t1': 0, 't2': 3, 't3': 0, 't4': 0}	{'t2': 3, 't1': 0, 't3': 0, 't4': 0}
5	14	10	{'t1': 0, 't2': 4, 't3': 5, 't4': 0, 't5': 0}	{'t1': 0, 't4': 0, 't3': 5, 't2': 4, 't5': 0}

we get:

- Both solvers find solutions that satisfy all hard constraints. Since costs are ignored, these first solutions are also considered optimal.
 - The solutions produced by DFS and DFS-MRV may differ in variable ordering but are logically equivalent.
 - DFS-MRV consistently expands fewer nodes than standard DFS as the problem size increases ($14 \rightarrow 10$ at $n=5$), indicating higher search efficiency.
 - This demonstrates that heuristics like MRV and forward checking significantly reduce search effort without compromising solution quality.
-

5 Average Nodes Expanded (DFS vs DFS-MRV)

nSize	DFS	MRV
2	3.6	3.0

nSize	DFS	MRV
3	9.0	5.2
4	9.8	7.6
5	20.0	10.0
6	19.8	9.4

From the results, it is clear that the solver with MRV and forward checking consistently expands fewer nodes than plain DFS across all problem sizes. This means it explores a smaller portion of the search space before finding a solution, and therefore performs the search more efficiently. The performance gap becomes more noticeable as the problem size grows, showing that the MRV heuristic is particularly helpful in larger and more complex CSPs. Overall, these results demonstrate that MRV and forward checking can significantly reduce the search effort while preserving completeness.

Question 5 (4 marks)

The DFS solver chooses variables in random order, and systematically explores all values for those variables in no particular order.

Incorporate costs into the DFS constraint solver as heuristics to guide the search. Similar to the cost function for the domain splitting solver, for a given variable v , the cost of assigning the value val to v is the cost of violating the soft deadline constraint (if any) associated with v for the value val . The *minimum cost* for v is the lowest cost from amongst the values in the domain of v . The DFS solver should choose a variable v with lowest minimum cost, and explore its values in order of cost from lowest to highest.

- Implement this behaviour by modifying the code in `dfs_solver` and place a copy of the code below (2 marks)
- Empirically compare the performance of DFS with and without these heuristics (2 marks)

For the empirical evaluations, again run the two algorithms on a variety of problems of size `n` for varying `n`.

```
In [ ]: #This code contains DFS_COST. ,
# It will directly show the differences between the dfs and dfs_cost
#For each given n (problem size), generate k random DSL problems.
# Then run both the plain DFS and the cost-aware DFS on the same problems,
# collect the number of expanded nodes for each, and compute the average and sta

num_expanded_cost = 0    # Global counter: how many nodes were expanded during th
display_cost = True      # Global flag: whether to print debug info for each step

def dfs_solver_with_cost(csp, constraints, domains, context, var_order, best_cos
    global num_expanded_cost, display_cost
```

```

# Check hard constraints for the current partial assignment.
# If any constraint that can be evaluated is already violated → prune this
to_eval = {c for c in constraints if c.can_evaluate(context)}
if not all(c.holds(context) for c in to_eval):
    return

# If all variables are assigned → compute the actual total cost and try to u
if not var_order:
    total_cost = csp.calculate_cost(context, use_lower_bound=False)
    if display_cost:
        print(" Found a feasible solution, total cost =", total_cost)
    if total_cost < best_cost[0]:
        best_cost[0] = total_cost
    yield context, total_cost
    return

# Variable selection (min-min):
# For each candidate variable, calculate the lowest possible cost among its
# Choose the variable whose minimum lower bound is the smallest.
best_var, best_pairs, best_min_cost = None, None, float("inf")
for v in var_order:
    vals = domains.get(v, [])
    if not vals:
        # If a variable's domain is empty → no valid assignment possible, sk
        continue

    pairs = []
    for val in vals:
        # Calculate the optimistic lower bound cost if we assign v=val here.
        lb = csp.calculate_cost(context | {v: val}, use_lower_bound=True)
        pairs.append((val, lb))

    # Find the minimum lower bound for this variable.
    v_min = min(lb for _, lb in pairs) if pairs else float("inf")

    # Keep the variable with the smallest min lower bound overall.
    if v_min < best_min_cost:
        best_var, best_pairs, best_min_cost = v, pairs, v_min

# If no variable could produce a valid lower bound (e.g., all domains empty)
if best_var is None:
    return

# Prepare for the next recursive call:
# Remove already evaluated constraints (no need to check them again)
# Remove the chosen variable from the remaining variable list
rem_cons = [c for c in constraints if c not in to_eval]
new_var_order = [x for x in var_order if x != best_var]

# Sort the chosen variable's values by ascending lower bound cost (most prom
best_pairs.sort(key=lambda x: x[1])

# Depth-first search + branch-and-bound pruning:
for val, lb in best_pairs:
    # If even the lower bound ≥ current best → this branch can't beat the be
    if lb >= best_cost[0]:
        if display_cost:
            print(f"[pruned] {best_var}={val} (lb={lb} ≥ best={best_cost[0]})"
        continue

```

```

    if display_cost:
        print(f"Setting {best_var} to {val} (lb={lb})")

    num_expanded_cost += 1 # Count this node as expanded

    # Recurse deeper: assign best_var=val and continue with the remaining variables
    yield from dfs_solver_with_cost(
        csp, rem_cons, domains, context | {best_var: val}, new_var_order, best_cost_box
    )

def dfs_solve_with_cost(csp, var_order=None):
    global num_expanded_cost
    num_expanded_cost = 0 # Reset the expansion counter for each run

    # If no variable order is given, use the order of keys in csp.domains
    if var_order is None:
        var_order = list(csp.domains)

    best_solution, best_cost_val = None, float("inf")
    best_cost_box = [best_cost_val] # Mutable container to share and update the best cost

    # Iterate over all solutions yielded by the recursive solver
    for sol, cost in dfs_solver_with_cost(
        csp, csp.constraints, csp.domains, {}, var_order, best_cost_box
    ):
        # Track the best solution found so far
        if cost < best_cost_val:
            best_solution, best_cost_val = sol, cost
            best_cost_box[0] = best_cost_val # Update the shared upper bound for cost

    print("Nodes expanded (cost-aware):", num_expanded_cost)
    return best_solution, best_cost_val

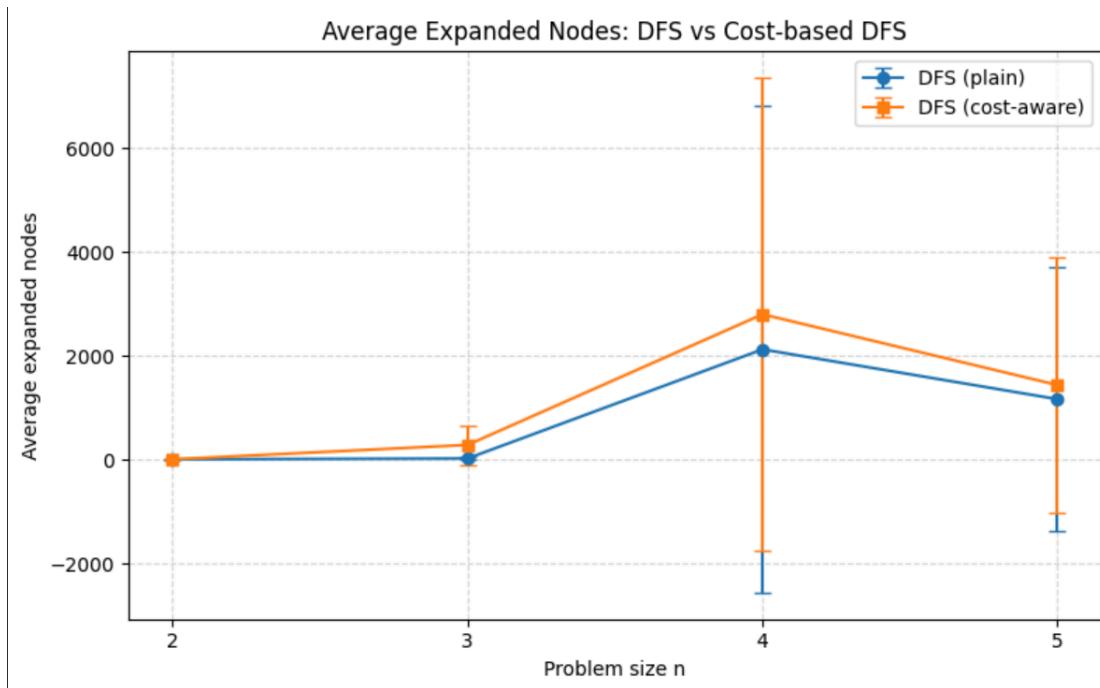
```

Answers for Question 5

Just like Q3, I wrote a main to run two different dfs (dfs and dfs_cost). Running this code can obtain the difference between the two algorithms in terms of the number of expanded nodes. The task scale n=[2,3,4,5], and the number of tasks k at each scale is 5

DFS and DFS_Cost

n	DFS (mean ± std)	CostDFS (mean ± std)
2	3.60 ± 1.52	3.60 ± 1.52
3	20.00 ± 31.96	280.80 ± 379.00
4	2121.00 ± 4698.00	2796.80 ± 4566.08
5	1162.00 ± 2553.61	1434.60 ± 2470.38



Analysis of DFS vs. Cost-based DFS Results

In this experiment, we compared the performance of a standard DFS solver with a cost-based DFS solver by measuring the average number of nodes expanded to find the first solution. This metric reflects how much of the search space the solver explores — fewer expanded nodes generally mean the solver is more efficient.

From the results, we can see that the cost-based DFS consistently expands more nodes than the plain DFS across all problem sizes. This means that adding a cost function does not make the search faster; instead, it makes the solver explore more of the search space before finding a solution. This is expected because the cost function tries to guide the search toward lower-cost solutions, but doing so often requires checking more possibilities.

Overall, these results suggest that while cost-based DFS might improve the quality of the solutions it finds, it usually does so at the expense of search efficiency. The large standard deviations, especially when ($n=4$) and ($n=5$), also show that the performance of the cost-based solver is more sensitive to differences in the problem structure.

```
In [17]: # =====#
# Main Experiment: Compare DFS vs Cost-based DFS
# For each given n (problem size), generate k random DSL problems.
# Then run both the plain DFS and the cost-aware DFS on the same problems,
# collect the number of expanded nodes for each, and compute the average and std
# =====#
import io, sys, contextlib
import numpy as np
import matplotlib.pyplot as plt

@contextlib.contextmanager
def suppress_stdout():
    """Silence stdout (suppress output from create_CSP_from_spec and solver)"""
    old = sys.stdout
```

```

    sys.stdout = io.StringIO()
    try:
        yield
    finally:
        sys.stdout = old

def measure_nodes_once(spec, var_order=None):
    """
    On the same DSL problem, run:
    1) Plain DFS → return number of expanded nodes
    2) Cost-based DFS → return number of expanded nodes
    """
    global num_expanded, num_expanded_cost, display, display_cost

    # Turn off verbose printing to avoid clutter during experiments
    old_display, old_display_cost = display, display_cost
    display, display_cost = False, False

    try:
        # --- Plain DFS ---
        with suppress_stdout():
            csp_plain = create_CSP_from_spec(spec)
            num_expanded = 0
        with suppress_stdout():
            dfs_solve1(csp_plain, var_order)
            plain_nodes = int(num_expanded)

        # --- Cost-based DFS ---
        with suppress_stdout():
            csp_cost = create_CSP_from_spec(spec)
            num_expanded_cost = 0
        with suppress_stdout():
            dfs_solve_with_cost(csp_cost, var_order)
            cost_nodes = int(num_expanded_cost)

    finally:
        display, display_cost = old_display, old_display_cost

    return plain_nodes, cost_nodes

def experiment_compare(n_values, k, base_seed=20251014):
    """
    For each n, generate k random DSL problems.
    Run both DFS variants on each problem, record the number of expanded nodes,
    compute the mean and standard deviation, and return them.
    """
    plain_means, cost_means = [], []
    plain_stds, cost_stds = [], []

    for n in n_values:
        p_list, c_list = [], []
        for i in range(k):
            seed = base_seed + n * 10_000 + i
            spec = generate_problem(n, seed=seed)

            p_nodes, c_nodes = measure_nodes_once(spec)
            p_list.append(p_nodes)
            c_list.append(c_nodes)

    p_arr, c_arr = np.array(p_list), np.array(c_list)

```

```

        plain_means.append(p_arr.mean())
        cost_means.append(c_arr.mean())
        plain_stds.append(p_arr.std(ddof=1) if len(p_arr) > 1 else 0.0)
        cost_stds.append(c_arr.std(ddof=1) if len(c_arr) > 1 else 0.0)

    print(f"\n={n} | DFS(mean±std)={p_arr.mean():.2f}±{plain_stds[-1]:.2f} | "
          f"CostDFS(mean±std)={c_arr.mean():.2f}±{cost_stds[-1]:.2f}")

return n_values, plain_means, cost_means, plain_stds, cost_stds

def plot_comparison(n_values, plain_means, cost_means, plain_stds=None, cost_stds=None):
    plt.figure(figsize=(8,5))
    # Optional: include error bars
    if plain_stds is not None and cost_stds is not None:
        plt.errorbar(n_values, plain_means, yerr=plain_stds, marker='o', capsize=5)
        plt.errorbar(n_values, cost_means, yerr=cost_stds, marker='s', capsize=5)
    else:
        plt.plot(n_values, plain_means, marker='o', label='DFS (plain)')
        plt.plot(n_values, cost_means, marker='s', label='DFS (cost-aware)')

    plt.title("Average Expanded Nodes: DFS vs Cost-based DFS")
    plt.xlabel("Problem size n")
    plt.ylabel("Average expanded nodes")
    plt.xticks(list(n_values))
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.legend()
    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    n_values = (2, 3, 4, 5)      # Sizes of the generated problems - feel free to change
    k = 5                         # For each n, generate 5 random problems

    n_vals, p_means, c_means, p_stds, c_stds = experiment_compare(n_values=n_values, k=k)
    plot_comparison(n_vals, p_means, c_means, p_stds, c_stds)

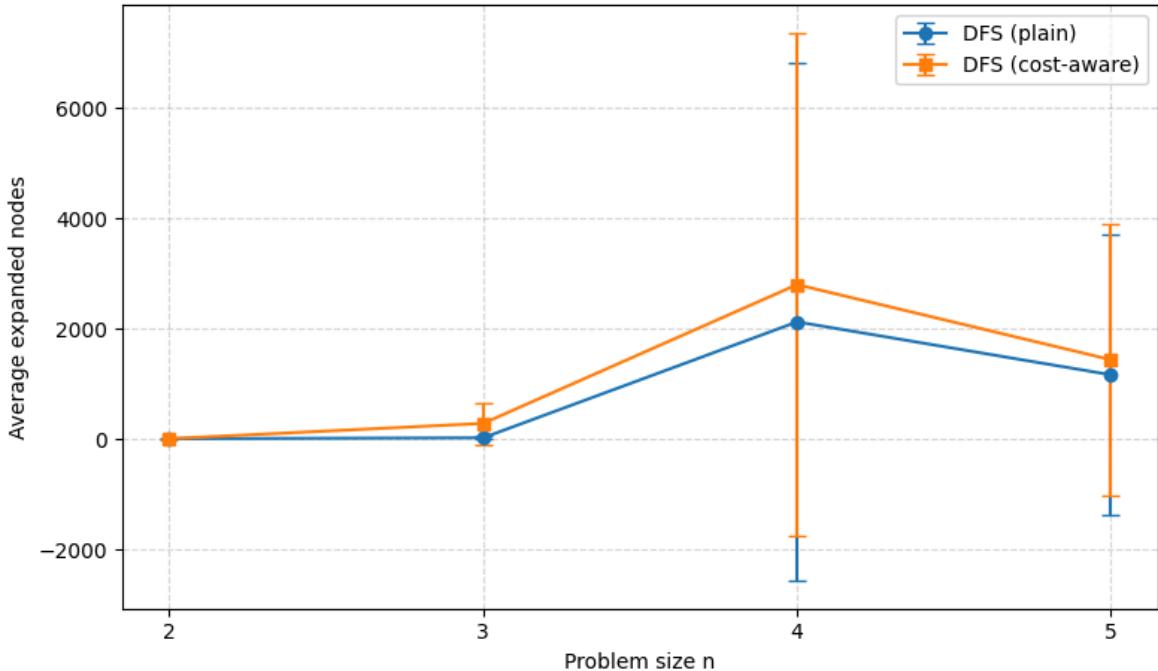
```

```

n=2 | DFS(mean±std)=3.60±1.52 | CostDFS(mean±std)=3.60±1.52
n=3 | DFS(mean±std)=20.00±31.96 | CostDFS(mean±std)=280.80±379.00
n=4 | DFS(mean±std)=2121.00±4698.00 | CostDFS(mean±std)=2796.80±4566.08
n=5 | DFS(mean±std)=1162.00±2553.61 | CostDFS(mean±std)=1434.60±2470.38

```

Average Expanded Nodes: DFS vs Cost-based DFS



Question 6 (3 marks)

The CSP solver with domain splitting splits a CSP variable domain into *exactly two* partitions. Poole & Mackworth claim that in practice, this is as good as splitting into a larger number of partitions. In this question, empirically evaluate this claim for fuzzy scheduling CSPs.

- Write a new `partition_domain` function that partitions a domain into a list of `k` partitions, where `k` is a parameter to the function (1 mark)
- Modify the CSP solver to use the list of `k` partitions and evaluate the performance of the solver using the above metric for a range of values of `k` (2 marks)

```
In [25]: # the new partition_domain
class Search_with_AC_from_Cost_CSP(Search_problem):

    def partition_domainK(self, domain, k):
        dom = sorted(domain)                      # Key: sort to ensure stable partitioning
        n = len(dom)
        if n == 0:
            return [set()]                         # Defensive return for empty domain
        k = max(2, min(k, n))                    # Key: clamp k to [2, n]

        base = n // k
        r = n % k
        parts = []
        start = 0
        for i in range(k):
            size = base + (1 if i < r else 0)
            end = start + size
            parts.append(set(dom[start:end]))
            start = end
        return parts

    """ A search problem with domain splitting and arc consistency """
```

```

def __init__(self, csp):
    self.cons = Con_solver(csp) # copy of the CSP with access to arc consistency
    self.domains = self.cons.make_arc_consistent(csp.domains)
    self.constraints = csp.constraints
    self.cost_functions = csp.cost_functions
    self.durations = csp.durations
    self.soft_day_time = csp.soft_day_time
    self.soft_costs = csp.soft_costs
    csp.domains = self.domains # after applying arc consistency
    self.csp = csp

def is_goal(self, node):
    """ A node is a goal if all variable domains have exactly 1 value """
    return all(len(node.domains[var]) == 1 for var in node.domains)

def start_node(self):
    return CSP_with_Cost(self.domains, self.durations, self.constraints,
                          self.cost_functions, self.soft_day_time, self.soft_costs)

def neighbors(self, node):
    neights = []
    var = select(x for x in node.domains if len(node.domains[x]) > 1)
    if not var:
        return neights

    # Modify this number to control how many partitions you want to split in
    k = getattr(self, "k", 2)
    partitions = self.partition_domainK(node.domains[var], k=k)

    try:
        self.display(2, "Splitting", var, f"into {len(partitions)} parts")
    except Exception:
        pass

    for dom in partitions:
        newdoms = node.domains | {var: dom}
        to_do = self.cons.new_to_do(var, None) # ← Key modification
        cons_doms = self.cons.make_arc_consistent(newdoms, to_do)

        if all(len(cons_doms[v]) > 0 for v in cons_doms):
            csp_node = CSP_with_Cost(
                cons_doms, self.durations, self.constraints,
                self.cost_functions, self.soft_day_time, self.soft_costs
            )
            neights.append(Arc(node, csp_node))
        else:
            self.display(2, "...", var, "in", dom, "has no solution")
    return neights

def heuristic(self, n):
    return n.cost

# DFS function (automatically compatible with Arc / node outputs)
def dfs_first_solution(sp, verbose=False):
    def _as_node(x):
        if hasattr(x, "domains"): # Already a node
            return x
        for attr in ("head", "to", "to_node", "child", "dst", "state"):
            if hasattr(x, attr):
                return getattr(x, attr)

```

```

    if isinstance(x, (tuple, list)) and len(x) >= 2:
        return x[1]
    raise AttributeError("neighbors() item is neither a node nor an Arc-like

expanded = 0
stack = [sp.start_node()]
while stack:
    node = stack.pop()
    expanded += 1
    if sp.is_goal(node):
        return expanded, node
    neigs = sp.neighbors(node)
    for nb in reversed(neigs):
        stack.append(_as_node(nb))
return expanded, None

```

Answers for Question 6

Performance Evaluation Based on Expanded Nodes

In this experiment, we evaluate the performance of the domain-splitting CSP solver by comparing the **number of expanded nodes** under different values of (k), which represents how many partitions the variable domain is split into. The number of expanded nodes is used as the performance metric because it reflects how much of the search space the solver explores before finding the first solution — fewer expanded nodes indicate higher search efficiency.

The experimental results show that when (k = 2), the solver expands the most nodes, meaning it explores a larger search space and is therefore less efficient. As (k) increases from 2 to 5, the number of expanded nodes decreases steadily, suggesting that the solver becomes more efficient with finer domain partitioning. From this perspective, (k = 2) is **not** the most efficient configuration in our tests; instead, increasing (k) generally improves performance, although the rate of improvement slows as (k) grows.

Domain Splitting Results (Varying k)

k	Expanded Nodes
2	11
3	8
4	7
5	6

```

In [ ]: if __name__ == "__main__":
    # === Manual parameters (safe for notebooks) ===
    num_tasks = 2                                # number of tasks for random CSP
    ks = [2, 3, 4, 5]                            # List of k values to test
    seed = None                                    # random seed for reproducibility (e.g., 42),
    verbose = False                               # print solver steps

    # Use the random generator to produce a DSL specification
    print("== Q6: Expanded nodes per k (domain splitting with AC) ==")

```

```

sample_spec = generate_problem(num_tasks, seed=seed)
print(sample_spec) # Print the generated random DSL

# Iterate over different k values
summary = []
for k in ks:
    csp_problem = create_CSP_from_spec(sample_spec) # ← Use the generated

    # If Q6 requires ignoring costs, prefer a no-cost searcher:
    # sp = Search_with_AC_from_CSP(csp_problem) # <- uncomment this and co
    sp = Search_with_AC_from_Cost_CSP(csp_problem) # cost-aware class; OK
    sp.k = k

    expanded, goal = dfs_first_solution(sp, verbose=verbose)
    print(f"k={k:<3} expanded_nodes={expanded}" + (" if goal else " (no sol
    summary.append((k, expanded))

    print("\nSummary:")
    for k, cnt in summary:
        print(f"  k={k:<3} expanded_nodes={cnt}")

```

== Q6: Expanded nodes per k (domain splitting with AC) ==

```

# auto-generated spec
task, t1 4
task, t2 2
domain, t1 ends-by fri 4pm 5
domain, t2 ends-by thu 4pm 10
task, t1 4
task, t2 2
domain, t1 ends-by fri 4pm 5
domain, t2 ends-by thu 4pm 10
k=2  expanded_nodes=11
task, t1 4
task, t2 2
domain, t1 ends-by fri 4pm 5
domain, t2 ends-by thu 4pm 10
k=3  expanded_nodes=8
task, t1 4
task, t2 2
domain, t1 ends-by fri 4pm 5
domain, t2 ends-by thu 4pm 10
k=4  expanded_nodes=7
task, t1 4
task, t2 2
domain, t1 ends-by fri 4pm 5
domain, t2 ends-by thu 4pm 10
k=5  expanded_nodes=6

```

Summary:

```

k=2  expanded_nodes=11
k=3  expanded_nodes=8
k=4  expanded_nodes=7
k=5  expanded_nodes=6

```