Weekly Menu PDDL Project

This repository provides two PDDL domains and corresponding problem definitions for a weekly menu planning task, alongside helper executables and scripts for generating and evaluating plans.

Repository Structure

```
PRAC3IA_PDDL/         # Project root
├── binaries/          # Platform executables and dependencies
│   ├── ff.exe         # Fast-Forward planner executable
│   ├── metricff.exe   # Metric version of Fast-Forward planner
│   └── cygwin1.dll    # Cygwin runtime library required on Windows
│
├── domain/            # PDDL domain definitions
│   ├── domain.pddl         # Basic (non-metric) weekly-menu domain
│   └── domain_metric.pddl  # Metric (numeric) extension of the domain
│
├── problems/          # Problem instances organized by extension
│   ├── basic/             # Extension 0–3 test problems
│   │   └── menu-test1.pddl
│   ├── ext1/              # Extension 1: "used" predicate tests
│   ├── ext2/              # Extension 2: previous-day-type tests
│   ├── ext3/              # Extension 3: forced-dish tests
│   ├── ext4/              # Extension 4: numeric-fluent (calories)
│   └── ext5/              # Extension 5: cost-optimization tests
│
├── scripts/           # Helper scripts and generated output
│   ├── generator/       # Script(s) to automatically produce problem files
│   ├── run/             # Script(s) to invoke planners on domains/problems
│   ├── dump_project.sh  # Script to dump file contents for review
│   └── data.txt         # Last output from `dump_project.sh`
│
├── .gitignore         # Specifies files and dirs ignored by Git
└── README.md           # This overview and setup instructions
```

Why this layout?

- **binaries/** keeps all external executables in one place, avoiding clutter in the root directory.
- **domain/** separates PDDL domain definitions from problem instances.
- **problems/** organizes test cases by feature extensions (ext1–ext5) for modular testing.
- **scripts/** holds utility scripts (generation, execution, dumping) and their outputs, making automation easy to find and update.
- **.gitignore** prevents temporary files (e.g. `data.txt`) and binaries from being committed unless desired.

With this structure, you can quickly locate or update any part of the planning pipeline, from domain models to test instances and execution tools.

Running & Executing Problem Instances

Before running any planner, ensure the `binaries/` folder is on your PATH:

```
export PATH="$(pwd)/binaries:$PATH"
```

Non-metric (basic) domains

To solve a basic problem (extensions 0–3), run:

```
ff.exe -o domain/domain.pddl -f problems/basic/menu-test1.pddl > basic_plan.txt
```

Metric (numeric) domains

To solve a metric problem (extensions 4–5), run:

```
metricff.exe -o domain/domain_metric.pddl -f problems/ext4/your_problem.pddl >
metric_plan.txt
```