Processamento em Nuvem

Você está trabalhando em uma grande empresa que virtualiza soluções na nuvem. Isso é tudo muito legal e inovador, mas tem problemas como qualquer empresa e você agora tem que resolver um deles: seus clientes querem processar grandes conjuntos de tarefas, que precisam ser feitas na ordem certa e levam tempos diferentes para serem realizadas. Se eles comprarem apenas um processador da sua nuvem, é claro que o tempo para completar tudo será a soma de todos os tempos das tarefas, mas quanto tempo demora se eles comprarem dois proessadores? E três? E quatro? E cento e quarenta e dois? Você está cansado de fazer continhas à mão para responder esse tipo de pergunta e resolveu automatizar o cálculo.

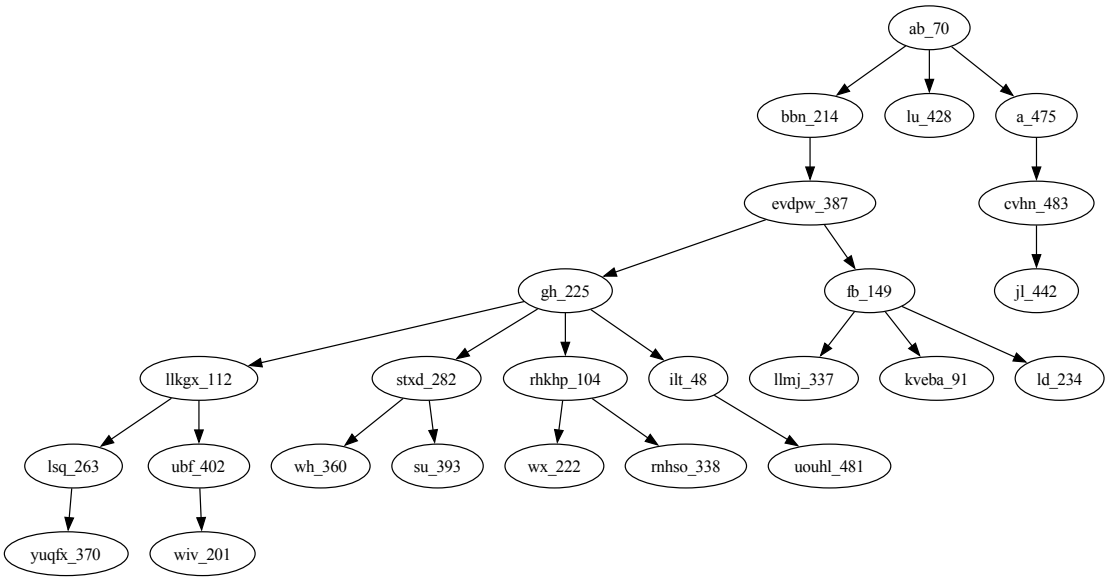
Você recebe dos clientes um esquema das tarefas que devem ser processadas: é uma longa lista de nomes de tarefas como na listagem abaixo, onde

```
blabla_213 -> tititi_53
```

significa que a tarefa `blabla_213` deve ser realizada antes que a tarefa `tititi_53` possa ser iniciada.

Tentando desenhar todas as conexões que existem na listagem abaixo, você termina com um desenho como o que está na sequência. Você sabe que o número que faz parte do nome de cada tarefa indica quanto tempo ela leva para ser realizada.

```
# Proc 4
lsq_263 -> yuqfx_370
llkgx_112 -> lsq_263
llkgx_112 -> ubf_402
gh_225 -> llkgx_112
gh_225 -> stxd_282
gh_225 -> rhkhp_104
gh_225 -> ilt_48
evdpw_387 -> gh_225
evdpw_387 -> fb_149
ubf_402 -> wiv_201
stxd_282 -> wh_360
stxd_282 -> su_393
bbn_214 -> evdpw_387
rhkhp_104 -> wx_222
rhkhp_104 -> rnhso_338
fb_149 -> llmj_337
fb_149 -> kveba_91
fb_149 -> ld_234
ab_70 -> bbn_214
ab_70 -> lu_428
ab_70 -> a_475
ilt_48 -> uouhl_481
a_475 -> cvhn_483
cvhn_483 -> jl_442
```



Você tem algumas informações extras:

- A linha inicial indica quantos processadores serão usados para as tarefas.

```
# Proc n
```

- Só um processador pode trabalhar numa tarefa, e a tarefa é encerrada sem interrupção.
- Diversos processadores podem trabalhar em tarefas ao mesmo tempo para terminar o mais rápido possível.
- Quando uma tarefa está livre para ser realizada por um processador ocioso (se houver), este é imediatamente alocado para o trabalho.
- Quando existem várias tarefas que podem ser escolhidas, você oferece duas políticas de escolha:
  - Política MIN: um processador é alocado para a tarefa que leva **menos tempo** (2178 unidades de tempo neste exemplo).
  - Política MAX: um processador é alocado para a tarefa que leva **mais tempo** (2337 unidades de tempo neste exemplo).

Você deve escrever o programa para encontrar o tempo de realização das tarefas de acordo com cada uma das políticas, para que o seu cliente possa escolher qual será usada, depois deve testá-lo com os casos de teste disponibilizados no Moodle e finalmente entregar o código-fonte e um relatório contando:

- Que problema está sendo resolvido e como foi modelado;
- Como foi o processo de solução, apresentando exemplos e algoritmos;
- Os resultados dos casos de teste;
- Análise de complexidade do(s) algoritmo(s) implementado(s);
- Conclusões.