

EP2 de MAC0422

Gabriel Capella
Luís Felipe de Melo

Terça-feira, 18 de outubro de 2016.

Problema

Temos que simular uma das provas do ciclismo, a perseguição por equipe. No programa, temos 2 equipes, e cada uma possui n ciclistas. A pista possui d metros. Cada time começa em um lado oposto da pista. Durante a corrida, apenas dois ciclistas podem estar lado a lado no circuito. Vence a equipe cujo terceiro ciclista ultrapassa o terceiro ciclista da outra equipe ou quando o terceiro ciclista de uma equipe termina 16 voltas primeiro.

Implementação

Implementação

- Cada ciclista é implementado com uma thread `ciclista`.
- Implementamos a pista como quatro vetores de tamanho $2*d$ (logo, cada ciclista ocupa duas posições no vetor):

Pista interna (`round = 0`)

Pista interna (`round = 1`)

Pista externa (`round = 0`)

Pista externa (`round = 1`)

- Fizemos essa escolha porque as velocidades podem variar, no modo `v`. Então, quando o ciclista está a 30 km/h, ele se move uma posição no vetor. Quando está a 60 km/h, move-se duas.

Implementação

- Declaramos uma variável `round`, que indica para qual pista o ciclista deve ir. A iteração referente a isso fica assim:

Cada ciclista vai para a pista interna do `round` diferente do atual. Se a posição de destino já estiver ocupada, ele tenta ir para a pista externa (ultrapassar). Caso a externa esteja ocupada, vai ficar uma posição atrás. As threads são sincronizadas e cada uma delas verifica as condições de término e aleatoriedade da corrida. Então, repetimos os passos.

Considerações

Considerações

- O custo da ultrapassagem (sair da pista interna e ir para a externa) é o mesmo custo de ir para a frente.
- Cada ciclista é autônomo, ele sabe as regras e decide se ultrapassa ou não.
- A corrida começa com todos os ciclistas fora da pista e os ciclistas da equipe A são inseridos nas posições 0 e 1 (já que em nossa implementação os ciclistas ocupam duas posições de vetor) do vetor e os ciclistas da equipe B são inseridos nas posições d e $d+1$.
- Sobre o ciclista que “segura” os outros: quando um ciclista passar na linha de chegada, ele vai ver se existe algum colega de equipe com número de voltas maior do que o dele em sua frente e que tenha a velocidade de 30km/h. Se ninguém tiver ele sorteia a velocidade, se alguém tiver ele vai andar a 30 km/h.

Barreiras

Barreiras

- Fizemos a nossa própria biblioteca de barreiras.
- A principal razão disso é que as barreiras da `pthread.h` não funcionariam direito quando um ciclista quebra. Quando ele quebra, sua thread é destruída, e o número de threads é alterado.

Probabilidade de quebra

Probabilidade de quebra

- Temos no enunciado que a probabilidade de quebra é 10% e 4 rodadas.
- Vamos considerar que ela seja uniforme, ou seja, a probabilidade de quebrar em movimento será:

$$P(q) = \frac{10\% * v}{d * 8}$$

- Onde v é a velocidade, que é 1 caso 30 km/h ou 2, caso contrário.
- Exemplo: Um ciclista está andando a 60 km/h. A probabilidade de que ele quebre a cada passo da iteração em uma pista de 400 m é de 0.00125.
- Vamos observar que o ciclista anda e quebra, não começa uma volta quebrado, mas pode terminar assim.

Probabilidade de quebra

- Em nosso código, utilizamos o tempo como raiz da função random.
- Além disso, temos a seguinte fórmula:

$$P(q) = \frac{speed}{80 * d}$$

- Note que duas vezes a distância máxima é igual à máxima posição que um ciclista pode estar, portanto o 40 da nossa fórmula.

Entrada

```
$ ./ep2 d n [v|u] [-v]
```

- d: número inteiro que representa o comprimento em metros da pista.
- n: número de ciclista em cada equipe.
- v: usado para definir simulações com velocidades aleatórias a cada volta.
- u: define simulações com velocidades uniformes de 60 km/h.
- -v: opção para mostrar tudo, utilizado para produzir saída para o terminal.

Saída

Testes