Tematy projektów zespołowych

1. Zadaniem programu o nazwie `scade` jest wyznaczenie wszystkich możliwych kaskadowych rozpadów cząstek elementarnych i ich graficzne przedstawienie na podstawie danych zawartych w pliku `snowmass2.spc` (plik ten znajduje się na GitHubie). Rozpadem kaskadowym nazywamy cały ciąg reakcji prowadzący od danej cząstki niestabilnej do cząstek stabilnych, które dalej się już nie rozpadają. Przykładowo niech cząstka a rozpada się na cząstki b_1 oraz c z prawdopodobieństwem $p_1 = 0.91$ i na cząstki b_2 oraz c z prawdopodobieństwem $p_2 = 0.02$. Z kolei cząstka b_1 rozpada się na x i z z prawdopodobieństwem $p_b = 0.07$, natomiast c na y i w z prawdopodobieństwem $p_c = 0.86$. Wreszcie z rozpada się na Z i γ z prawdopodobieństwem $p_z = 0.39$. Cząstki b_2, x, y, w, Z oraz γ są stabilne i nie rozpadają się dalej. Reakcje te można symbolicznie zapisać jako

$$a \to b_1 c, \quad a \to b_2 c, \quad b_1 \to xz, \quad c \to yw, \quad z \to Z\gamma. \tag{1}$$

W wyniku kolejnych rozpadów uzyskujemy następujące rozpady kaskadowe i odpowiadające im prawodpodobieństwa

$$a \to b_1 c \to xzc \to xzyw \to xZ\gamma yw \quad (p_1 p_b p_c p_z \approx 0.021), \tag{2}$$
$$a \to b_2 c \to b_2 yw \quad (p_2 p_c = 0.017), \tag{3}$$
$$b_1 \to xz \to xZ\gamma \quad (p_b p_z = 0.027), \tag{4}$$
$$c \to yw \quad (p_c = 0.86). \tag{5}$$

Plik tekstowy `snowmass2.spc` zawiera informacje o masach cząstek i ich rozpadach w jednym z możliwych scenariuszy rozszerzenia Modelu Standardowego. W pliku tym cząstki oznaczane są tzw. kodami PDG. Przykładowo, z następującego fragmentu tego pliku

```
BLOCK MASS  # Mass Spectrum
# PDG code           mass       particle
        36     1.51103826E+03   # A
        37     1.51343530E+03   # H+
   1000001     1.55931152E+03   # ~d_L
   2000001     1.55240630E+03   # ~d_R
   1000002     1.55431328E+03   # ~u_L
```

można odczytać, że cząstka `A` ma kod PDG 36 i masę 1.511×10^3 GeV, cząstka `H+` ma kod 37 i masę 1.513×10^3 GeV, cząstka `~d_L` ma kod 1000001 i masę 1.559×10^3, GeV, itd. Antycząstki oznaczane są kodami takimi jak cząstki, ale z dodatkowym znakiem minus; np. antycząstka cząstki `36` jest oznaczana kodem `-36`. Korzystając z tych konwencji jeden z kanałów rozpadu cząstki `1000021` można zapisać w następującym formacie:

```
#         PDG            Width
DECAY   1000021     2.61914150E-03   # gluino decays
#          BR         NDA      ID1       ID2       ID3
    1.17953941E-02    3     1000023      6        -6   # BR(~g -> ~chi_20 t  tb)
```

W tym przypadku cząstka o kodzie 1000021 rozpada się na 3 cząstki (NDA oznacza ilość produktów rozpadu) 1000023, 6 oraz antycząstkę cząstki 6 czyli na -6:

$$1000021 \rightarrow 1000023\ 6\ -6 \tag{6}$$

Proces ten można również zapisać korzystając z tradycyjnych symboli w następujący sposób:

$$\tilde{}g \rightarrow \tilde{}chi\_20\ t\ tb \tag{7}$$

Prawdopodobieństwo tego rozpadu podane jest w kolumnie oznaczonej przez BR. W tym przypadku jest to około 1.18×10^{-2}. W tym samym pliku można również odnaleźć infomację dotyczącą rozpadów cząstek o kodach 6 i 1000023:

```
#         PDG            Width
DECAY         6     1.56194983E+00   # top decays
#          BR         NDA      ID1       ID2
     1.00000000E+00    2        5        24   # BR(t ->  b    W+)
#         PDG            Width
DECAY   1000023     1.00000000E-03   # neutralino2 decays
#          BR         NDA      ID1       ID2
     1.00000000E+00    2     1000022        23   # BR(~chi_20 -> ~chi_10   Z )
```

Dzięki tym danym można skonstruować następujący rozpad kaskadowy:

$$1000021 \rightarrow 1000023\ 6\ -6 \rightarrow 1000022\ 23\ 6\ -6 \rightarrow 1000022\ 23\ 5\ 24\ -5\ -24 \tag{8}$$

i obliczyć jego prawdopodobieństwo: 1.18×10^{-2}. Rozważając pozostałe kanały rozpadu cząstki 1000021 a następnie powtarzając całą procedurę dla wszystkich cząstek można wyznaczyć wszystkie rozpady kaskadowe. Jeżeli dla którejś cząstki nie są wypisane żadne rozpady oznacza to, że cząstka ta jest stabilna. Przykładem takich cząstek mogą być cząstki o kodach 1, 11 badź 1000022.

Struktura programu scade powinna umożliwiać dostarczanie danych wejściowych poprzez podanie kodu PDG cząstki, dla której chcemy wyznaczyć wszystkie rozpady kaskadowe, bezpośrednio w linii poleceń w terminalu (stdin). Przykładowo dla cząstki 1000021 wywołanie programu powinno wyglądać następująco

```
$ .\scade --PDG=1000021 --BR=0.001 --input=snowmass2.txt --output=wyniki.txt
          --graph=rozpady.eps
```

gdzie

- flaga --PDG określa kod PDG cząstki, dla której liczone są rozpady kaskadowe; --PDG=ALL powinno wymuszać wypisanie do pliku lub na standardowe wyjście wszystkich możliwych rozpadów kaskadowych,
- flaga --BR ustala minimalne prawdopodobieństwo, poniżej którego rozpady kaskadowe nie są wypisywane do pliku lub na stdout (wartość domyślna 0),
- flaga --input określa plik z danymi wejściowymi,
- flaga --output określa plik z wynikami (wartość domyślna stdout)
- flaga --graph=rozpady.eps określa plik w formacie eps z reprezentacją graficzną rozpadów kaskadowych (niezdefiniowanie flagi --graph powoduje, że reprezentacja graficzna nie jest rysowana)

Zarówno w przypadku wypisania wyników na standardowe wyjście jak i przy zapisie do pliku wyniki powinny być przedstawione w następującym formacie:

```
{1000021,{1000022,23,5,24,-5,-24},1.1795E-02}
  .
  .
  .
```

Ponadto powinny być one posortowane malejąco ze względu na ostatnią kolumnę zawierającą prawdopodobieństwa rozpadów. Dodatkowo program powinien generować reprezentację graficzną całych rozpadów kaskadowych (np. w formie diagramu) i zapisywać ją do pliku `rozpady.eps`.

3. Program o nazwie `cgsu2` ma obliczać tzw. współczynniki Clebscha-Gordana

$$\mathtt{cg}(\mathtt{j},\mathtt{m},\mathtt{j1},\mathtt{m1},\mathtt{j2},\mathtt{m2}) \tag{9}$$

zwiazane z nieprzywiedlnymi reprezentacjami grupy $SU(2)$ i wykonywać na nich określone poniżej operacje prowadzące do wyznaczenia trzech nieujemnych liczb rzeczywistych `cl1`, `cl2a` oraz `cl2b`.

Niech liczby $\mathtt{j},\mathtt{j1},\mathtt{j2}$ będą postaci $k/2$, gdzie $k = 0, 1, 2, 3\ldots$ Dla każdego zestawu trzech takich liczb $\{\mathtt{j},\mathtt{j1},\mathtt{j2}\}$ program `cgsu2` ma wyznaczać trzy liczby rzeczywiste $\mathtt{cl1}(\mathtt{j},\mathtt{j1},\mathtt{j2})$, $\mathtt{cl2a}(\mathtt{j},\mathtt{j1},\mathtt{j2})$ oraz $\mathtt{cl2b}(\mathtt{j},\mathtt{j1},\mathtt{j2})$. Pierwsza z nich jest zdefiniowana w następujący sposób:

$$\mathtt{cl1}(\mathtt{j},\mathtt{j1},\mathtt{j2}) = \sum_{\mathtt{m1}}\sum_{\mathtt{m2}} |\mathtt{cg}(\mathtt{j},\mathtt{m},\mathtt{j1},\mathtt{m1},\mathtt{j2},\mathtt{m2})|^2, \tag{10}$$

gdzie sumowanie jest po $\mathtt{m1} = -\mathtt{j1},\ -\mathtt{j1}+1,\ldots,\mathtt{j1}-1,\mathtt{j1}$ oraz analogicznie po $\mathtt{m2} = -\mathtt{j2}$, $-\mathtt{j2}+1,\ldots,\mathtt{j2}-1,\mathtt{j2}$. Program ma obliczać liczbę $\mathtt{cl1}(\mathtt{j},\mathtt{j1},\mathtt{j2})$ oddzielnie dla każdej z wartości $\mathtt{m} = -\mathtt{j},-\mathtt{j}+1,\ldots,\mathtt{j}-1,\mathtt{j}$ oraz sprawdzać, że dla każdej z nich liczba $\mathtt{cl1}(\mathtt{j},\mathtt{j1},\mathtt{j2})$ jest taka sama. Z kolei $\mathtt{cl2a}(\mathtt{j},\mathtt{j1},\mathtt{j2})$ oraz $\mathtt{cl2b}(\mathtt{j},\mathtt{j1},\mathtt{j2})$ są zdefiniowane przez:

$$\mathtt{cl2a}(\mathtt{j},\mathtt{j1},\mathtt{j2}) = \sum_{\mathtt{m1}}\sum_{\mathtt{m2}}\sum_{\mathtt{m'}}\sum_{\mathtt{m2'}} |\mathtt{cg}(\mathtt{j},\mathtt{m},\mathtt{j1},\mathtt{m1},\mathtt{j2},\mathtt{m2})|^2 |\mathtt{cg}(\mathtt{j1},\mathtt{m1},\mathtt{j},\mathtt{m'},\mathtt{j2},\mathtt{m2'})|^2 \tag{11}$$

oraz

$$\mathtt{cl2b}(\mathtt{j},\mathtt{j1},\mathtt{j2}) = \sum_{\mathtt{m1}}\sum_{\mathtt{m2}}\sum_{\mathtt{m'}}\sum_{\mathtt{m1'}} |\mathtt{cg}(\mathtt{j},\mathtt{m},\mathtt{j1},\mathtt{m1},\mathtt{j2},\mathtt{m2})|^2 |\mathtt{cg}(\mathtt{j2},\mathtt{m2},\mathtt{j},\mathtt{m'},\mathtt{j1},\mathtt{m1'})|^2. \tag{12}$$

Program powinien obliczać $\mathtt{cl2a}(\mathtt{j},\mathtt{j1},\mathtt{j2})$ oraz $\mathtt{cl2b}(\mathtt{j},\mathtt{j1},\mathtt{j2})$ oddzielnie dla każdej z wartości $\mathtt{m} = -\mathtt{j},-\mathtt{j}+1,\ldots,\mathtt{j}-1,\mathtt{j}$ oraz sprawdzać, że $\mathtt{cl2a}(\mathtt{j},\mathtt{j1},\mathtt{j2})$ i $\mathtt{cl2b}(\mathtt{j},\mathtt{j1},\mathtt{j2})$ nie zależą od $\mathtt{m}$.

Aby współczynniki $\mathtt{cg}(\mathtt{j},\mathtt{m},\mathtt{j1},\mathtt{m1},\mathtt{j2},\mathtt{m2})$ były niezerowe musi zachodzić następująca nierówność:

$$|\mathtt{j1}-\mathtt{j2}| \leqslant j \leqslant \mathtt{j1}+\mathtt{j2}. \tag{13}$$

Ponadto gdy $\mathtt{j1} > \mathtt{j2}$ i $\mathtt{m} > 0$ to współczynniki $\mathtt{cg}(\mathtt{j},\mathtt{m},\mathtt{j1},\mathtt{m1},\mathtt{j2},\mathtt{m2})$ mają postać:

$$\begin{aligned}
\mathtt{cg}(\mathtt{j},\mathtt{m},\mathtt{j1},\mathtt{m1},\mathtt{j2},\mathtt{m2}) \;=\; & \delta_{\mathtt{m},\mathtt{m1}+\mathtt{m2}} \sqrt{\frac{(2\mathtt{j}+1)(\mathtt{j}+\mathtt{j1}-\mathtt{j2})!(\mathtt{j}-\mathtt{j1}+\mathtt{j2})!(\mathtt{j1}+\mathtt{j2}-\mathtt{j})!}{(\mathtt{j1}+\mathtt{j2}+\mathtt{j}+1)!}} \\
& \times \sqrt{(\mathtt{j}+\mathtt{m})!(\mathtt{j}-\mathtt{m})!(\mathtt{j1}-\mathtt{m1})!(\mathtt{j1}+\mathtt{m1})!(\mathtt{j2}-\mathtt{m2})!(\mathtt{j2}+\mathtt{m2})!} \\
& \times \sum_{\mathtt{k}} \left[\frac{(-1)^{\mathtt{k}}}{\mathtt{k}!(\mathtt{j1}+\mathtt{j2}-\mathtt{j}-\mathtt{k})!(\mathtt{j1}-\mathtt{m1}-\mathtt{k})!(\mathtt{j2}+\mathtt{m2}-\mathtt{k})!} \right. \\
& \left. \times \frac{1}{(\mathtt{j}-\mathtt{j2}-\mathtt{m1}+\mathtt{k})!(\mathtt{j}-\mathtt{j1}-\mathtt{m2}+\mathtt{k})!} \right],
\end{aligned} \tag{14}$$

gdzie sumowanie przebiega po takich liczbach całkowitych k, że argument każdego wyrażenia z silnią jest nieujemny. Aby wyznaczyć współczynniki cg(j, m, j1, m1, j2, m2) dla m < 0 bądź j1 $<$ j2 należy skorzystać z relacji

$$\texttt{cg(j,m,j1,m1,j2,m2)} = (-1)^{\texttt{j-j1-j2}}\texttt{cg(j,-m,j1,-m1,j2,-m2)} \tag{15}$$

oraz

$$\texttt{cg(j,m,j1,m1,j2,m2)} = (-1)^{\texttt{j-j1-j2}}\texttt{cg(j,m,j2,m2,j1,m1)}. \tag{16}$$

Struktura programu cgsu2 powinna umożliwiać dostarczanie danych wejściowych na trzy sposoby:

a) Poprzez podanie trzech liczb {j, j1, j2} bezpośrednio w linii poleceń w terminalu (stdin). Przykładowo dla j = 0, j1 = 1/2, j2 = 1/2 wywołanie programu powinno wyglądać następująco

```
$ .\cgsu2 0 1/2 1/2
```

Program powinien wypisywać w terminalu (stdout) następujący komunikat z wynikiem (w podanym przykładzie jest to liczba 1):

```
cl1(0, 1/2, 1/2) = 1
cl2a(0, 1/2, 1/2) = 1
cl2b(0, 1/2, 1/2) = 1
```

b) Poprzez wczytanie pliku tekstowego dane.txt

```
$ .\cgsu2 dane.txt wyniki.txt
```

zawierającego dowolną liczbę zestawów j, j1, j2, np.

```
{0, 1/2, 1/2}
{1, 1/2, 1}
     .
     .
     .
{j, j1, j2}
     .
     .
     .
```

Program cgsu2 dla każdego zestawu {j, j1, j2} z pliku dane.txt powinien obliczać odpowiadające mu liczby cl1, cl2a oraz cl2b i zapisywać wyniki do pliku wyniki.txt w następujący sposób:

```
{{0, 1/2, 1/2}, 1,1,1}
{{1, 1/2, 1}, 0,0,0}
       .
       .
       .
{{j, j1, j2}, cl1, cl2a, cl2b}
       .
       .
       .
```

c) Poprzez podanie trzech liczb `j`, `j1` oraz `j2` w prostym interfejsie graficznym wywoływanym z linii poleceń poprzez komendę

```
$ .\cgsu2
```

Obliczenie liczb `cl1`$(j, j1, j2)$, `cl2a`$(j, j1, j2)$ oraz `cl2b`$(j, j1, j2)$ odpowiadających zestawowi $j, j1, j2$ powinno odbywać się po naciśnięciu odpowiedniego przycisku w interfejsie. Liczby `cl1`, `cl2a` oraz `cl2b` powinny pojawić się w oddzielnych polach interfejsu.

5. Zadaniem programu o nazwie `madlog` jest zarządzanie bazą danych, gromadzącą wyniki symulacji numerycznych prowadzonych na zdalnych komputerach. Program `madlog` powinien:

a) odczytywać w ustalonych odstępach czasu (np. co 1 min.) dane numerycznych gromadzone na zdalnych komputerach poprzez połączenie przez `ssh` z tymi komputerami (1. opcja) bądź odczytywać te danych z lokalnego komputera dzięki synchronizacji poprzez Dropboxa (2. opcja).

b) aktualizować bazę danych poprzez dopisywanie nowych wyników pobranych w trakcie realizacji punktu a),

c) przeszukiwać bazę danych na podstawie zapytań sformułowanych przez użytkownika,

d) umieszczać wybraną w punkcie c) część wyników wewnątrz kodu `html`, tak aby możliwe było ich odczytanie poprzez przeglądarkę `www`,

e) tworzyć graficzną reprezentację uzyskanych wyników numerycznych w postaci wykresu zapisanego w formacie `eps` i umieszczać odpowiedni link do takiego wykresu w w.w. kodzie `html`.

Aby zobrazować schemat działania programu `madlog` rozważmy następujący przykład. Na dwóch zdalnych komputerach oznaczonych jako $\{b, 4\}$ oraz $\{t, 8\}$, gdzie 4 i 8 to ilość rdzeni używanych podczas obliczeń, uruchomione są następujące procesy:

- na komputerze b proces `T1` oraz proces `T2`; w procesie `T1` wyznaczana jest wartość funkcji $f(x, y, z)$ dla $n_x = 25$ punktów $x = 1, 2, \ldots, 25$ przy ustalonym $y = 8$ i $z = 0.1$, natomiast w procesie `T2` wyznaczana jest wartość funkcji $f(x, y, z)$ dla $n_y = 50$ punktów $y = 1, 2, \ldots, 50$ przy ustalonym $y = -2$ i $z = 0.3$. Wyniki są zapisywane do plików tekstowych `T1.txt` oraz `T2.txt`. Plik `T1.txt` jest sformatowany następująco

$$\begin{array}{lll} 1 & f(1,8,0.1) & 2015:10:15:20:21:14 \\ 2 & f(2,8,0.1) & 2015:10:15:21:01:36 \\ 3 & f(3,8,0.1) & 2015:10:15:23:14:06 \\ . & & \\ . & & \\ . & & \\ k & f(k,8,0.1) & \tau_k \end{array} \tag{17}$$

gdzie τ_k oznaczają kolejne momenty (w formacie daty `yyyy:mm:dd:hh:mm:ss`), w których zostało ukończone obliczanie wartości funkcji $f(k, 8, 0.1)$. Analogiczny format ma plik `T2.txt`

- na komputerze t proces `T3`; w procesie tym wyznaczana jest wartość funkcji $f(x, y, z)$ dla $n_z = 10$ punktów $z = 1, 2, \ldots 10$ przy ustalonym $x = 0.73$ i $y = 1.4$. Wyniki są zapisywane

do pliku tekstowego `T3.txt` w następującej postaci:

$$\begin{array}{lll} 1 & f(0.73, 1.4, 1) & 2015:10:11:07:01:55 \\ 2 & f(0.73, 1.4, 2) & 2015:10:12:04:22:06 \\ 3 & f(0.73, 1.4, 3) & 2015:10:13:17:01:44 \\ . & & \\ . & & \\ . & & \end{array} \tag{18}$$

Załóżmy, że wszystkie trzy pliki `T1.txt`, `T2.txt`, `T3.txt` są zsynchronizowane poprzez Dropboxa z ich kopiami na lokalnym komputerze. Pierwszym zadaniem programu `madlog` jest skopiowanie danych z tych plików do lokalnej bazy danych `B`. Postać bazy nie jest z góry ustalona. Może ona wyglądać następująco:

$$\begin{aligned} \texttt{B} \;=\; \{ & \\ & \{\texttt{T1}, \{x, \{1, 25\}, 25, \texttt{BT1}\}, \{b, 4\}, \{2015:10:15:20:21:14, 28\texttt{h}34\texttt{m}, 0.87\}\}, \\ & \{\texttt{T2}, \{y, \{1, 50\}, 50, \texttt{BT2}\}, \{b, 4\}, \{2015:10:15:16:31:40, 29\texttt{h}04\texttt{m}, 0.96\}\}, \\ & \{\texttt{T3}, \{z, \{1, 10\}, 10, \texttt{BT3}\}, \{t, 8\}, \{2015:10:11:07:01:55, 109\texttt{h}22\texttt{m}, 1.0\}\}, \\ & \}, \end{aligned} \tag{19}$$

gdzie kolejne elementy pierwszego rekordu bazy oznaczają: nazwę procesu $\rightarrow$ `T1`, wielkość, która się zmienia w kolejnych krokach iteracji, jej zakres zmienności i liczbę kroków iteracji $\rightarrow \{x, \{1, 25\}, 25, \texttt{BT1}\}$, nazwę zdalnego komputera i liczbę użytych rdzeni $\rightarrow \{b, 4\}$, moment rozpoczęcia iteracji, całkowity czas, który upłynął od momentu rozpoczęcia iteracji oraz wskaźnik postępu pracy $\rightarrow \{2015:10:15:20:21:14, 28\texttt{h}34\texttt{m}, 0.87\}$ (0.0 oznacza, że iteracje nie zostały jeszcze rozpoczęte, a 1.0 oznacza, że wszystkie iteracje zostały już ukończone). Wreszcie `BT1` oznacza listę wyników procesu `T1`:

$$\texttt{BT1} = \{\{1, f(1, 8, 0.1)\}, \{2, f(2, 8, 0.1)\}, \ldots\}. \tag{20}$$

Program `madlog` powinnien umożliwiać przeszukiwanie bazy `B` ze względu na określone zapytanie np. o procesy, które nie są jeszcze ukończone badź o procesy liczone na komputerze t. Ponadto `madlog` powininen również zapisywać tak przygotowane dane w formie kodu `html`, tak aby możliwe było ich wyświetlenie w przeglądarce `html`. Ostatnim zadaniem programu `madlog` powinno być przedstawienie danych zawartych w listach `BT1`, `BT2`, `BT3` w postaci wykresów (wykonanych przy użyciu np. programu `gnuplot`) zależności funkcji $f(x, y, z)$ kolejno od zmiennych x, y oraz z.